НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики і Обчислювальної техніки

Кафедра Обчислювальної Техніки

«До захисту допущено»

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___» ___ 2020 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

за спеціальністю 121 «Інженерія програмного забезпечення»

на тему: «Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення»

Виконав:
студент IV курсу, групи ІП-62
МІРОШНИК Денис Валерійович _____

Керівник:
доцент, кандидат технічних наук.
ПОРЄВ Віктор Миколайович _____

Консультант з нормоконтролю:
Професор, доктор технічних наук
СІМОНЕНКО Валерій Павлович _____

Рецензент:
_____
_____ _____

Засвідчую, що в цьому дипломному проекті немає запозичень з праць іншихавторів без відповідних посилань
Студент _____

Київ — 2020

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет Інформатики і Обчислювальної Техніки

Кафедра Обчислювальної Техніки

Ступінь бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

за спеціальністю 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___» _____ 2019р.

ЗАВДАННЯ

на дипломну роботу студенту

Мірошнику Денису Валерійовичу

1. Тема роботи: «Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення»,
керівник роботи доцент, к.т.н. Порєв Віктор Миколайович,
2. Термін подання студентом роботи: «4» червня 2020 р.
3. Вихідні дані до роботи: розроблювана система повинна працювати з текстами і надавати оцінку їх подібності від 0 до 100
4. Зміст роботи: Виконати огляд та аналіз алгоритмів порівняння тексту, оглянути існуючі системи менеджменту програмного забезпечення, розробити сервіс менеджменту, який використовує систему захисту від створення дублікатів задач.
5. Перелік ілюстративного матеріалу: структурна схема системи, узагальнена схема роботи системи, блок-схема алгоритму системи
6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
/.	*Затвердження теми роботи*	*10.12.2019-15.12.2019*	
2.	*Вивчення та аналіз завдання*	*15.12.2019-15.03.2020*	
3.	*Розробка архітектури та загальної структури систем*	*15.03.2020-15.04.2020*	
4.	*Розробка структур окремих підсистем*	*15.04.2020-01.05.2020*	
5.	*Програмна реалізація системи*	*01.05.2020-15.05.2020*	
6.	*Оформлення пояснювальної записки*	*15.05.2020-25.05.2020*	
7.	*Захист програмного продукту*	*05.06.2020*	
8.	*Передзахист*	*20.06.2020*	
9.	*Захист*	*25.06.2020*	

Студент _____

(підпис)

Керівник роботи _____

(підпис)

АНОТАЦІЯ

Метою даної дипломної роботи є створення сервісу менеджменту розробки програмного забезпечення з використанням системи захисту від дублікатів задач.

У роботі проведено аналіз існуючих алгоритмів оцінки подібності текстів — відстань Жаккара, кластеризація методом к–середніх, косинус подібності, латентно-семантичний аналіз (Latent Semantic Indexing — LSI), латентне розміщення Діріхле та дивергенція Дженсена-Шеннона, варіаційний автокодувальник, універсальний кодувальник речення, manhattan LSTM (MaLSTM), BERT + Косинус подібності. Виконано огляд існуючих систем менеджменту програмного забезпечення – JIRA, Trello, GitHub Boards, Asana, Monday, Avaza, ClickUp, Redmine, Microsoft TFS, YouTrack. Описано інструменти розробки – Python, Django, SQLite. Описано розроблений веб-сервіс на базі наведених інструментів.

Ключові слова: сервіси відслідковування задач, дублікати текстів, розпізнавання тексту, порівняння текстів.

АННОТАЦИЯ

Целью данной работы является создание сервиса менеджмента разработки программного обеспечения с использованием системы защиты от дубликатов задач.

В работе проведен анализ существующих алгоритмов оценки сходства текстов – расстояние Жаккара, k-means, косинус сходства, латентно-семантический анализ (Latent Semantic Indexing - LSI), латентное размещения Дирихле и дивергенция Дженсена-Шеннона, вариационный автокодировщик, универсальный кодировщик предложения, manhattan LSTM (MaLSTM), BERT + Косинус сходства. Выполнен обзор существующих систем менеджмента программного обеспечения - JIRA, Trello, GitHub Boards, Asana, Monday, Avaza, ClickUp, Redmine, Microsoft TFS, YouTrack. Описаны инструменты разработки - Python, Django, SQLite. Описан разработанный веб-сервис на базе приведенных инструментов.

Ключевые слова: сервисы отслеживания задач, дубликаты текстов, распознавания текста, сравнение текстов.

ABSTRACT

The purpose of this thesis is to create a software management service using a system of protection against tasks duplication.

The analysis of existing algorithms for estimating the similarity of texts - Jacquard distance, clustering by k-means, cosine of similarity, latent-semantic analysis (LSI), latent Dirichlet placement and Jensen-Shannon divergence, variational autocode, manhattan LSTM (MaLSTM), BERT + Cosine similarity. An overview of existing software management systems - JIRA, Trello, GitHub Boards, Asana, Monday, Avaza, ClickUp, Redmine, Microsoft TFS, YouTrack. Described development tools - Python, Django, SQLite. The developed web service based on the given tools is described.

Keywords: task tracking services, duplicate texts, text recognition, text comparison.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІЛАЦ.467100.001 ВП	Відомість проекту	1	
3	А4	ІЛАЦ.467100.002 ТЗ	Технічне завдання	4	
4	А4	ІЛАЦ.467100.003 ПЗ	Пояснювальна записка	54	
5	А3	ІЛАЦ.467100.004 Д1	Схема принципова	1	
6	А3	ІЛАЦ.467100.005 Д2	Схема функціональна	1	
7	А3	ІЛАЦ.467100.006 Д3	Схема структурна	1	

					ІЛАЦ.467100.001 ВП			
Зм.	Арк.	№ документу	Підпис	Дата				
Розробив		Мірошник Д.В.			Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення Відомість дипломного проекту	Літ.	Аркуш	Аркушів
Перевірив		Порєв В.М.					6	1
						НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, гр. ІП-62		
Н. Контр.		Симоненко В.П.						
Затверд.								

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи

освітньо-кваліфікаційного рівня «бакалавр»

на тему: “ Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку курсу «Веб-програмування». Область застосування: практичне використання людьми в повсякденнному житті для планування задач.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр з інженерії програмного забезпечення», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи захисту від створення дублікатів задач у сервісах менеджменту виробництва програмного забезпечення.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування, публікації та наукові статті в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробляємого продукту

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Можливість динамічні конфігурації апаратної частини системи.
- Незалежність – система повинна мати програмну автономність і не залежати від встановленого програмного забезпечення.
- Швидка обробка даних з мінімальними похибками

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 98 та новіші версії
- Доступ до мережі Інтернет.
- Пошуковий браузер.

5.3. Вимоги до апаратної частини

- Комп’ютер на базі процесора Intel або AMD мінімальної потужності.
- Оперативної пам’яті не менше 512 Мбайт.
- Наявність набору сенсорів та мікрокомп’ютеру, налаштованих згідно вимогам

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	10.03.2020
Складання і узгодження технічного завдання	01.04.2020
Створення модулів системи, що розробляється	10.04.2020
Тестування окремих модулів системи	20.04.2020
Допрацювання, налагодження і виправлення помилок	04.05.2020
Оформлення документації дипломної роботи	17.05.2020

Пояснювальна записка
до дипломного проекту
на тему: «Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення»

ЗМІСТ

ВСТУП

Зі зростанням промисловості розробки програмного забезпечення та утворенням великих підприємств, організацій та команд виникає потреба у покращені засобів управління великими командами спеціалістів у галузі програмної інженерії та тестування програмного забезпечення.

Інструменти для команд розробників та тестувальників повинні забезпечувати можливість:

- Планування – створення та розподілення задач між командою, планування спринтів.
- Прозорості – визначення пріоритету, контексту, змісту та мети задач.
- Відслідковування – видимість етапу виконання задач та прогрес спринта.

Інструменти націлені на спрощення роботи менеджерів, скрам-майстерів, проект-координаторів, спрощення взаємодії команд розробників і тестувальників.

Актуальність роботи полягає в розробці доповнення до існуючих можливостей сервісів та інструментів менеджменту розробки програмного забезпечення, а саме доповнення сервісу системою захисту від створення дублікатів задач, багів.

Мета полягає у зменшенні ризику перевантаження системи інформацією, що повторюється та натомість перевикористовувати вже створені задачі та баги з ідентичним контекстом.

РОЗДІЛ 1. ОГЛЯД СИСТЕМ МЕНЕДЖМЕНТУ ВИРОБНИЦТВА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Оскільки проекти зростають за розмірами і складністю, обмеження у можливостях аркуша Excel при використанні таких для відстеження задач починають проявлятися дуже швидко. Коли так стається, то з'являється потреба у розгляді використання систем відстеження задач, щоб переконатися, що всі питання вирішуються вчасно.

Системи менеджменту задач, які зазвичай називають ITS – Issue Tracking System, - це інструменти або онлайн сервіси, які забезпечують можливість детального та гнучкого управління задачами.

Кінцевому користувачеві, який застосовує ці інструменти, вони в основному допомагають повідомити про проблему, а потім відстежувати, як вирішується проблема, а деякі такі сервіси навіть можуть показувати яка людина чи команда призначена для її вирішення. Що стосується розробника програмного забезпечення, то сервіс менеджменту дозволяє налаштувати інструмент відстеження відповідно до потреб та мати центральну систему координації на різних етапах розробки програмного забезпечення. Відмінне програмне забезпечення вважається доглянутим, якщо інструмент відстеження проблем постійно оновлюється. Більшість інструментів використовують невеликі картки-тікети, англійською "ticket", які є найменшою одиницею відстеження проблеми, що містить визначення проблеми, статус пріоритету та всі інші важливі пов'язані з тікетом дані.

В тікеті може бути все, що завгодно від повідомлення про помилку до запитання клієнта розробників.

Коли десять різних клієнтів повідомляють про десять різних проблем, викликаних одним багом, десять тікетів відкриваються в ITC і відстежуються, поки вони не будуть вирішені, як правило, з наступним оновленням застосунку та послідуючою перевіркою відзвітувавших клієнтів, що вони більше не мають жодних проблем.

1.1. Особливості систем менеджменту задач

Кожен тікет в ITS, як правило, містить кілька деталей, пов'язаних з ним. Деякі тікети можуть мати більшу актуальність, ніж інші, можливо, тому, що вони зачіпають більшість замовників або тому, що вони означають серйозну проблему, яку потрібно вирішити, щоб проект міг продовжувати роботу без проблем. Так само питання, що мають низький або нульовий пріоритет, щоб вказати, що вони повинні бути вирішені тоді, коли це дозволяє час після того, як більш пріоритетні завдання будуть вирішені.

Інші деталі включають все, від замовника, який відчуває проблему, до детального опису проблеми, що виникає, до спроб вирішення та іншої відповідної інформації.

Більшість застосунків ITS також дозволяють призначати відповідальними за завдання різних осіб, стежити за тим, як триває робота над завданням та скільки часу витрачається на них, забезпечують відповідність внутрішнім робочим процесам, здійснюють статистичний аналіз та автоматично генерують тікети на основі запитів замовника

Порівняно з більш традиційними системами менеджменту програмного виробництва, сучасні застосунки забезпечують підзвітність та необхідні інструменти для забезпечення вирішення питань належним чином, і вони часто інтегруються з іншими інструментами розробки програмного забезпечення.

Існує безліч таких систем та сервісів. Є популярні, якими користуються як великі так і малі виробництва програмного забезпечення, а також інші, які менш актуальні для широкого вжитку, але мають незамінні для когось власні особливості.

1.2. JIRA

Jira розробником якої є Atlassian це власницький інструмент управління проектами з широкими можливостями відстеження задач. Вона була вперше випущена у 2002 році, що робить її однією з найстаріших систем відстеження задач у світі, і її зараз використовують понад 75 000 клієнтів по всьому світу.

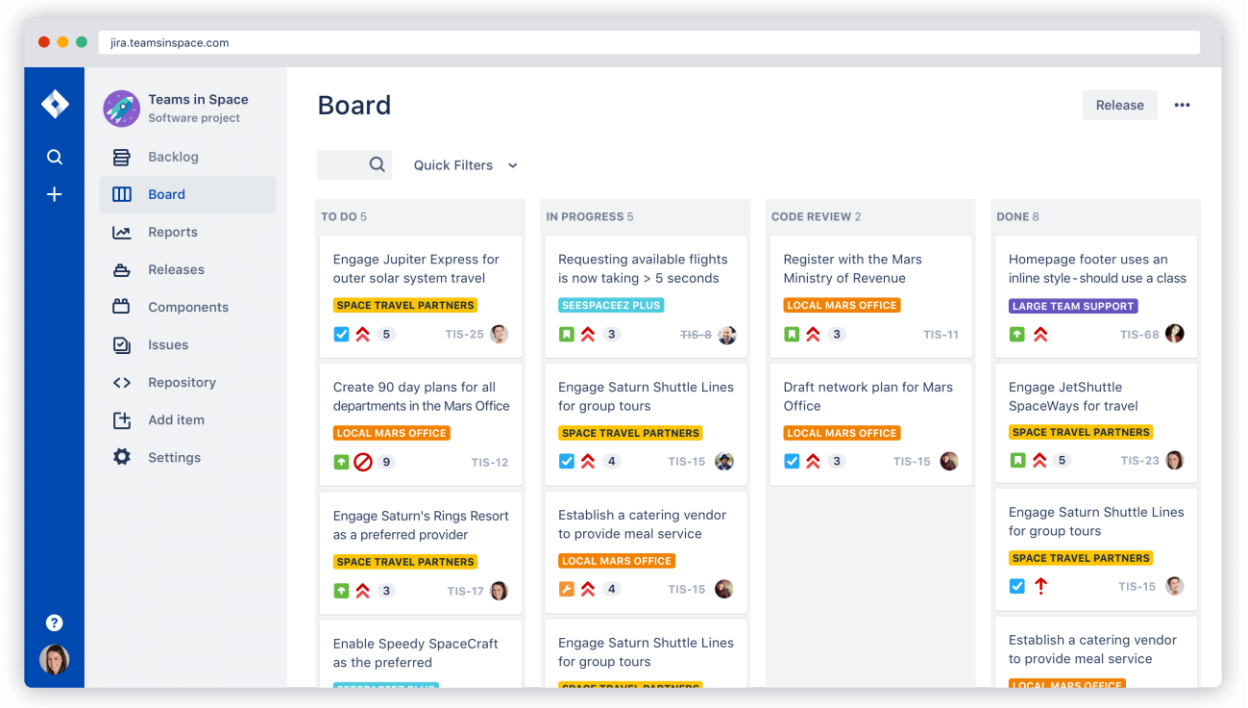


Рисунок 1.1 Інтерфейс сервісу JIRA

JIRA - це дуже багатогранний інструмент, саме тому деякі найбільші організації світу, включаючи Twitter, Skype та NASA, використовували його в певний час. З тієї самої причини JIRA може бути заплутаною і незрозумілою, не кажучи вже про те, що вона коштує тисячі доларів на рік для дуже великих команд.

1.3. Trello

Trello - це дуже популярний інструмент відстеження задач, який організовує проекти в дошки. В Trello можна відстежувати над чим іде робота, хто виконує або хто призначений на виконання і на якій стадії знаходиться той чи інший тікет. Це в основному біла дошка, наповнена добре заповненими та впорядкованими наліпками. Сервіс доступний навіть на мобільних пристроях.

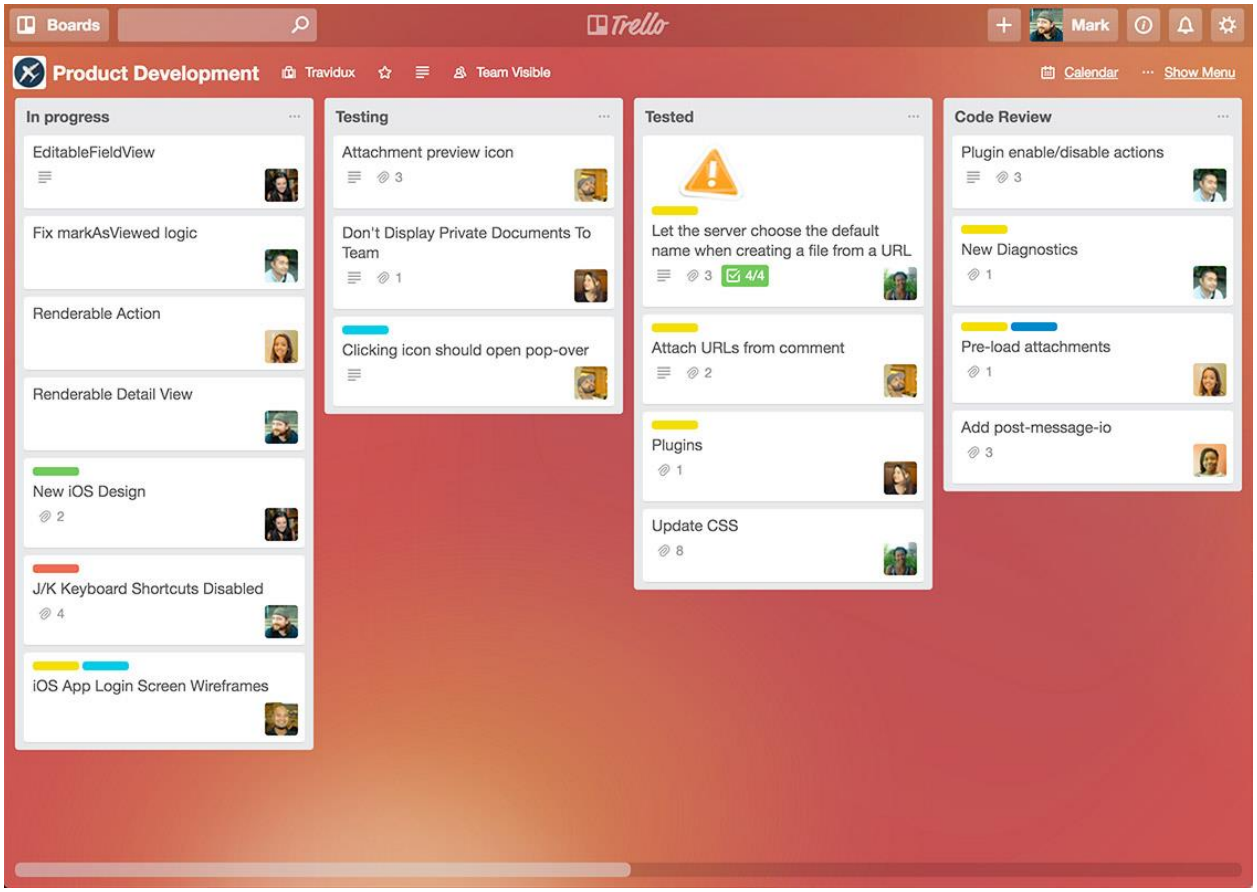


Рисунок 1.2 Інтерфейс сервісу Trello

1.4. GitHub Boards

GitHub Boards є офіційною системою менеджменту проектів GitHub і інструмент відстеження виконання задач, який уже стає улюбленим інструментом віддалених команд через швидку інтеграцію з вашим сховищем GitHub. Він розміщений у кожному сховищі, створеному на GitHub у розділі проектів. Ці проекти складаються із pull запитів, нотаток та задач, які можна організовувати як картки у стовпцях на ваш вибір, GitHub надає абсолютний контроль над дошкою проектів та має можливість автоматизувати кожну подію відстеження тікетів.

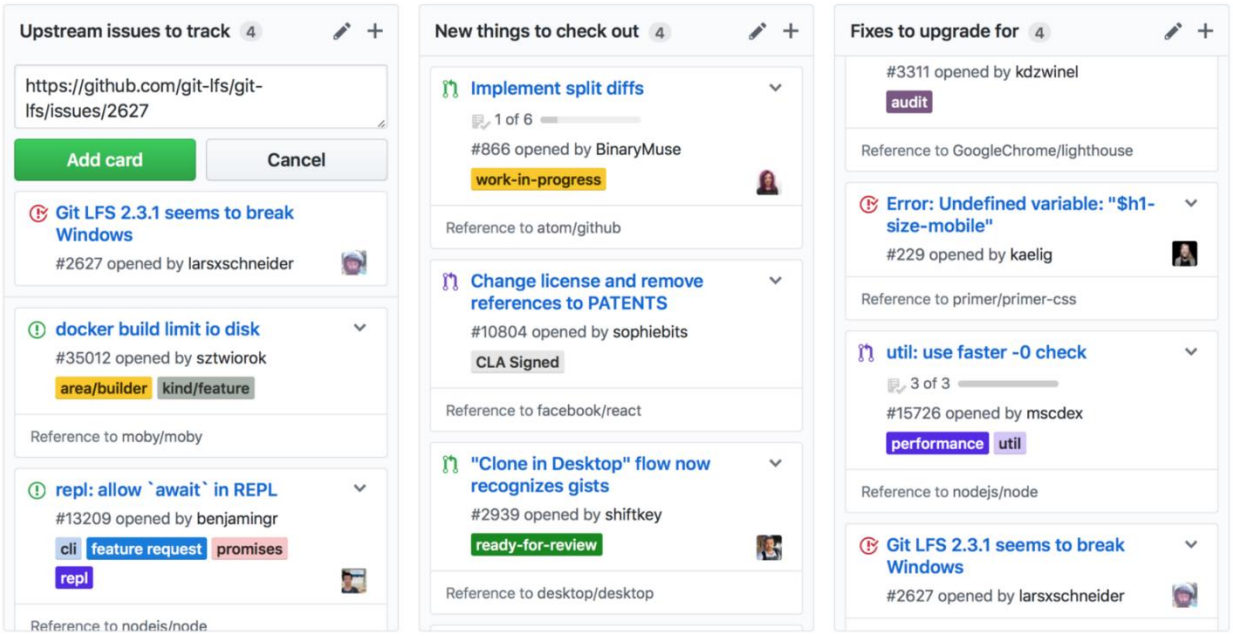


Рисунок 1.3 Інтерфейс сервісу GitHub Boards

1.5. Asana

Завдяки такому прекрасному інтерфейсу користувача Asana - це інструмент менеджменту проектів та відстеження задач, заснований співзасновником Facebook Дастіном Московіцем та Джастіном Розенштейнам, створеним для того, щоб допомогти командам організовувати, відстежувати та керувати багами та задачами. Asana може похвалитися дуже інтуїтивно зрозумілим користувацьким інтерфейсом. Можна навіть створити візуальні плани проектів, щоб побачити, як кожен крок відображається з часом, точно визначає ризики, усуває проблеми на шляху розвиту проекту навіть тоді, коли плани змінюються. Також є функція портфоліо, де можна стежити за всіма своїми репозиторіями в режимі реального часу. Asana також дає можливість призначити роботу учасникам команди, визначати та вказувати терміни та коментувати завдання, а також включає інструменти звітності, вкладення файлів, календарі та інші корисні функції.

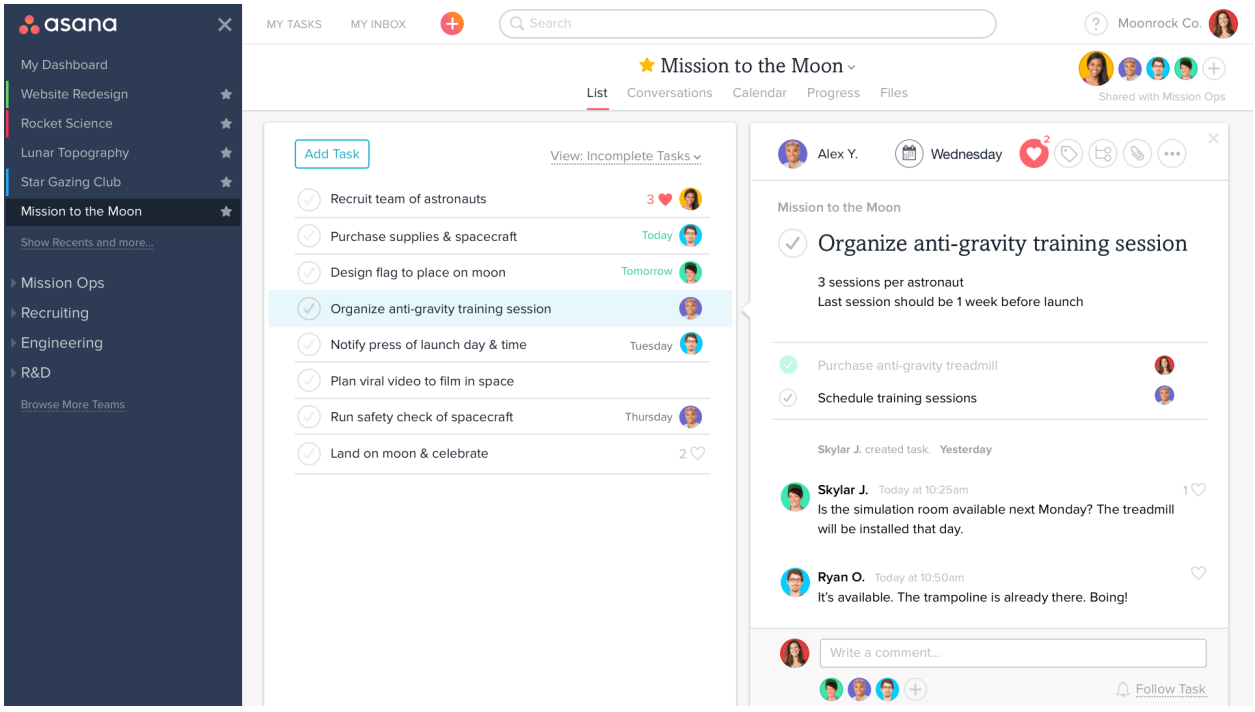


Рисунок 1.4 Інтерфейс сервісу Asana

1.6. Monday

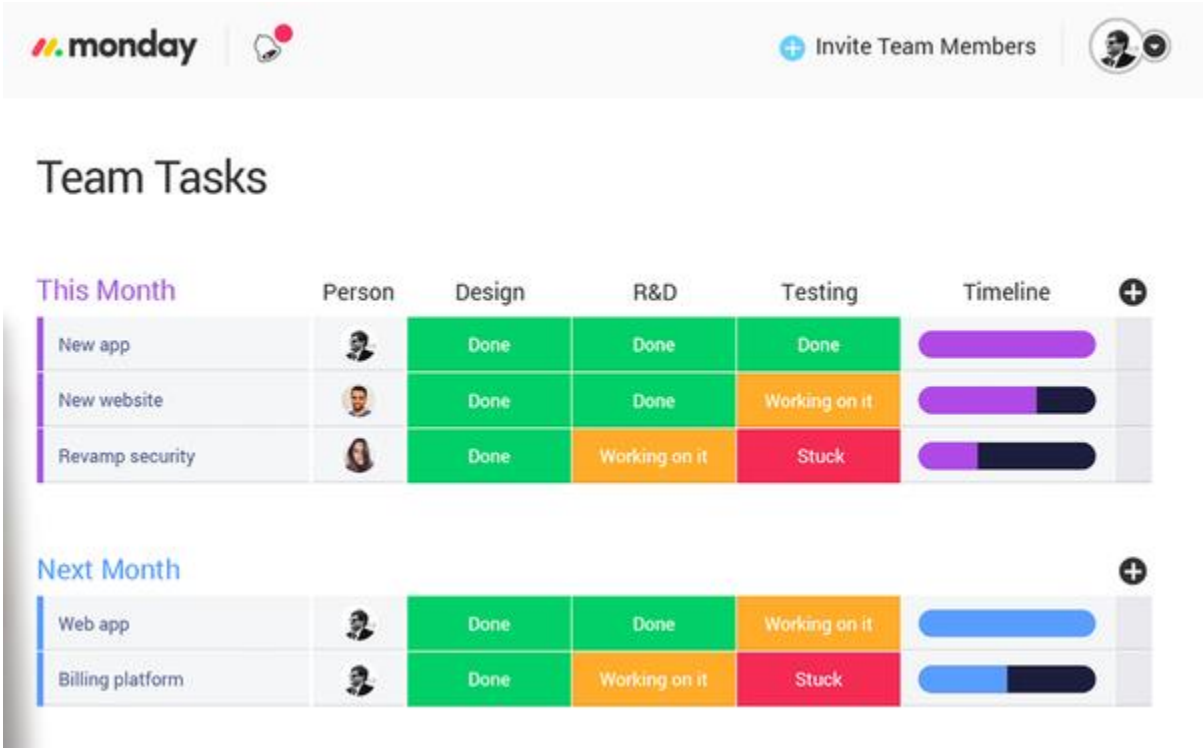


Рисунок 1.5 Інтерфейс сервісу Monday

Monday – це корисний і гнучкий онлайн сервіс інструмент, який спрощує спільної роботи команд керуванням робочим навантаженням, відстежуючи стан проектів та їх тікетів. Ця система включає засоби комунікації, інструменти простого візуального керування робочим процесом. Можна створювати і налаштовувати дошки для команд, сторінки з потрібними даними, тощо.

1.7. Avaza

Avaza – це система відслідковування задач, що включає функції управління проектами, планування ресурсів, онлайн-розклади, управління витратами, онлайн-рахунки, періодичні рахунки та інше. Avaza має дуже інтуїтивну інформаційну панель, а також інтегрується з сторонніми платформами для управління робочими процесами. Вона має потужні функції планування ресурсів із відстеженням часу та різною зручною візуалізацією.

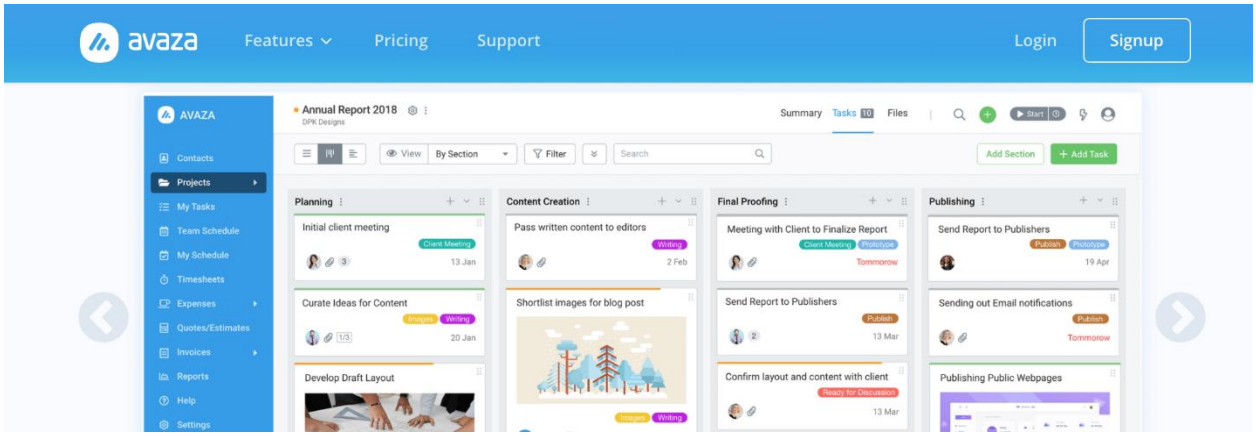


Рисунок 1.6 Інтерфейс сервісу Avaza

1.8. ClickUp

ClickUp виділяється майже з усіх інших сервісів управління та інструментів відстеження. За допомогою ClickUp можна отримувати різні перспективи огляду проекту та згодом інформаційні панелі для різних команд та їх завдань. Наприклад, у дизайнерів та розробників є абсолютно різні інформаційні панелі, а також окремі для керівників проектів. Він також має набір перспектив: списки, дошки, тощо, щоб давати змогу вибрати свою панель приладів.

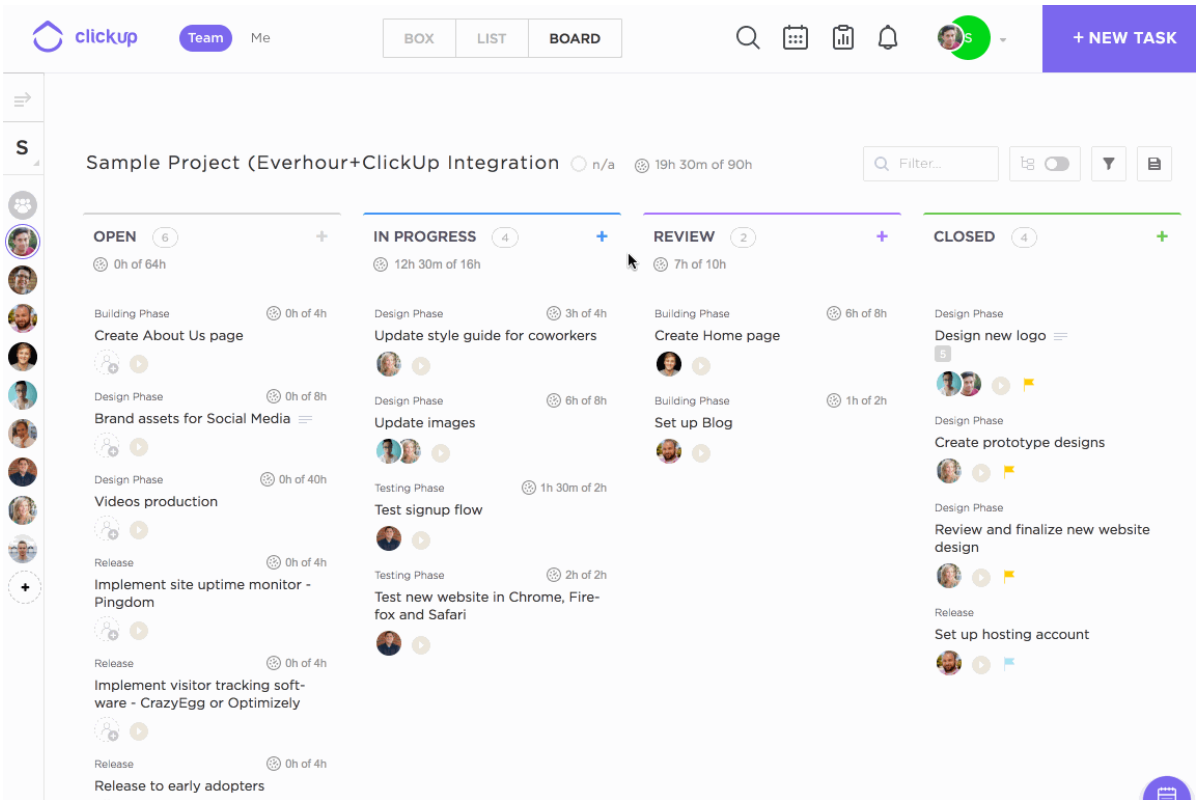


Рисунок 1.7 Інтерфейс сервісу Click up

1.9.Redmine

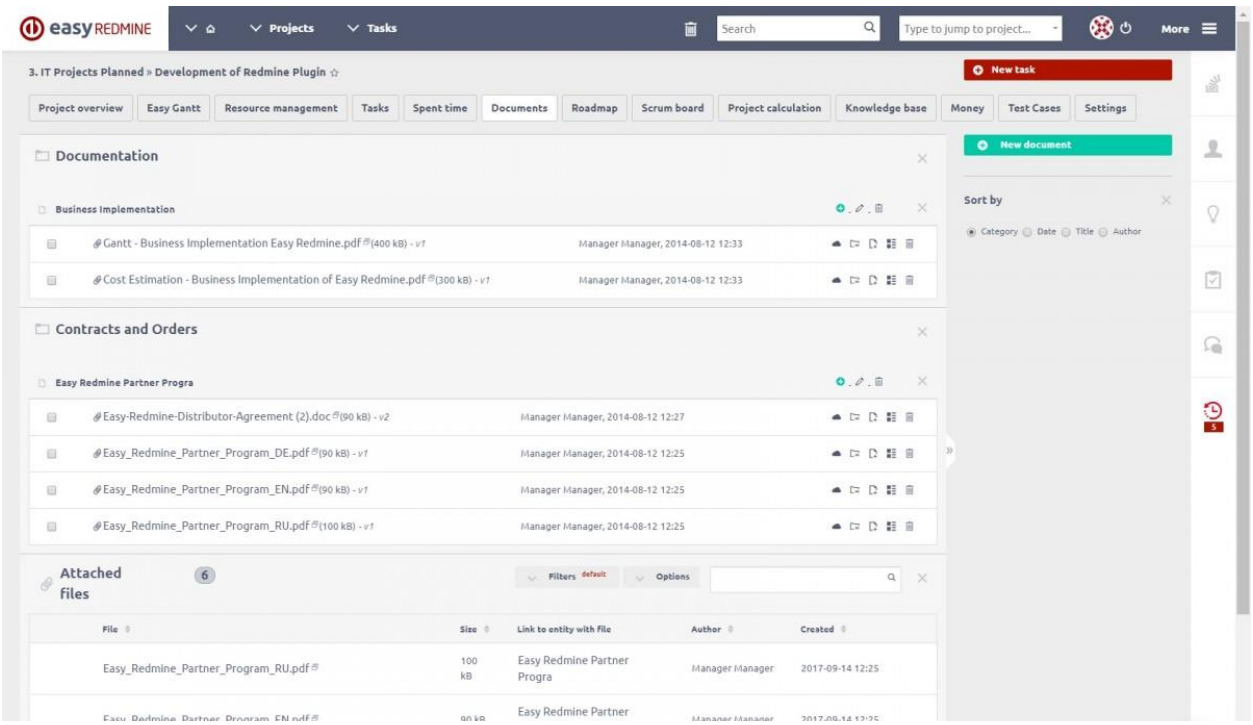


Рисунок 1.8 Інтерфейс сервісу Redmine

Redmine – це безкоштовний та з відкритим кодом онлайн сервіс менеджменту тікетів та проектів. Користувачі цього сервісу мають можливість управляти декількома проектами та їх підпроектами. Він оснащений довідковими проектами, їх форумами, відстеженням часу та гнучким контролем доступу на основі ролей. Також дозволяє візуалізувати дані про хід виконання проектів графіками. Сервіс постійно оновлюється та підтримується.

1.10. Microsoft TFS

Microsoft Team Foundation Server це продукт від компанії Microsoft, що дозволяє керувати кодом, тікетами, відгуками, управляти вимогами, проектами, автоматизувати збірки оновлень програмного забезпечення, тестування та відслідковування за новими оновленнями програмного забезпечення, яке розробляється з допомогою сервісу. Він охоплює весь життєвий цикл розробки та забезпечує можливості DevOps.

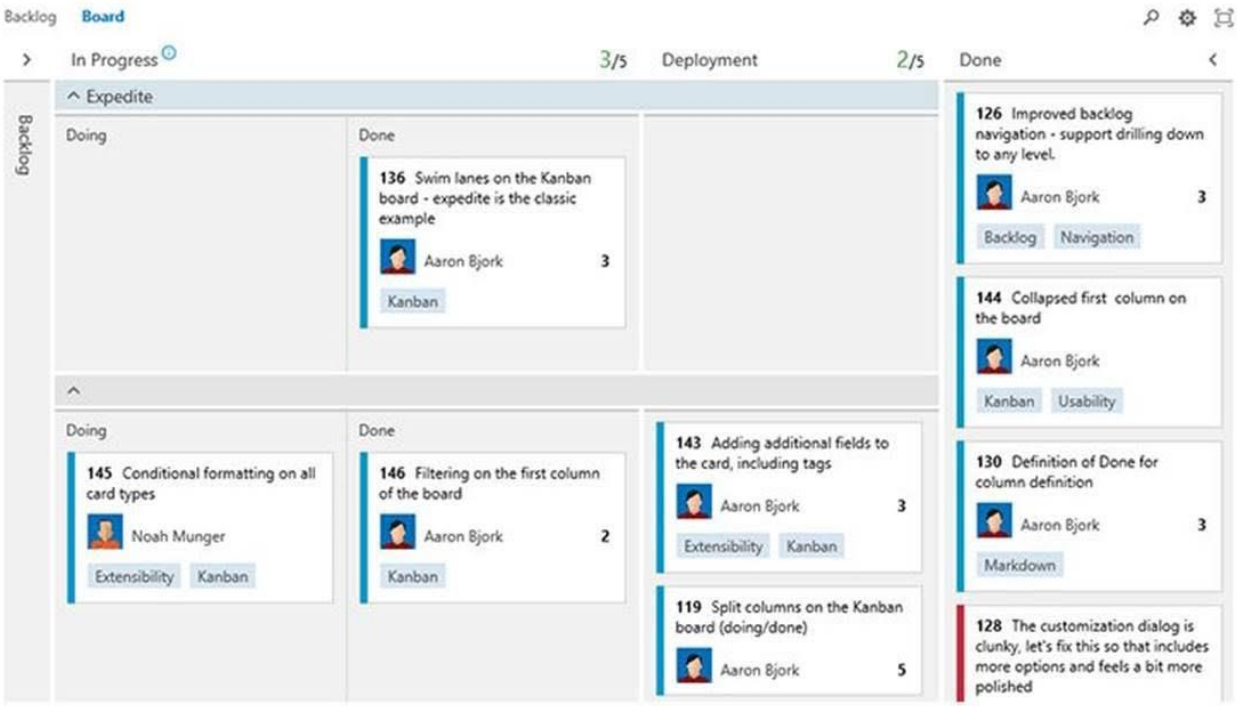


Рисунок 1.9 Інтерфейс сервісу Microsoft TFS

Це корисний набір інструментів, таких як Jira, а також має ще більш чітку інтеграцію з середовищем для розробки програмного забезпечення Visual Studio як Team Server.

1.11. YouTrack

Побудований командою в Jetbrains, Youtrack - це комерційний браузер для відстеження тікетів та програмного забезпечення для управління проектами. Інструмент управління проектами, який можна адаптувати до ваших процесів, щоб допомогти вам доставити чудові продукти. Відстежуйте завдання та помилки, плануйте спринти та випуски, створюйте робочі процеси та налаштовуйте YouTrack для своїх бізнес-процесів. Ніколи не змушуйте ваш процес знову відповідати меж інструменту. На відміну від інших трекерів випуску, YouTrack можна налаштувати під ваші потреби. Це дуже популярний варіант для спритних команд і безкоштовний для проектів з відкритим кодом.

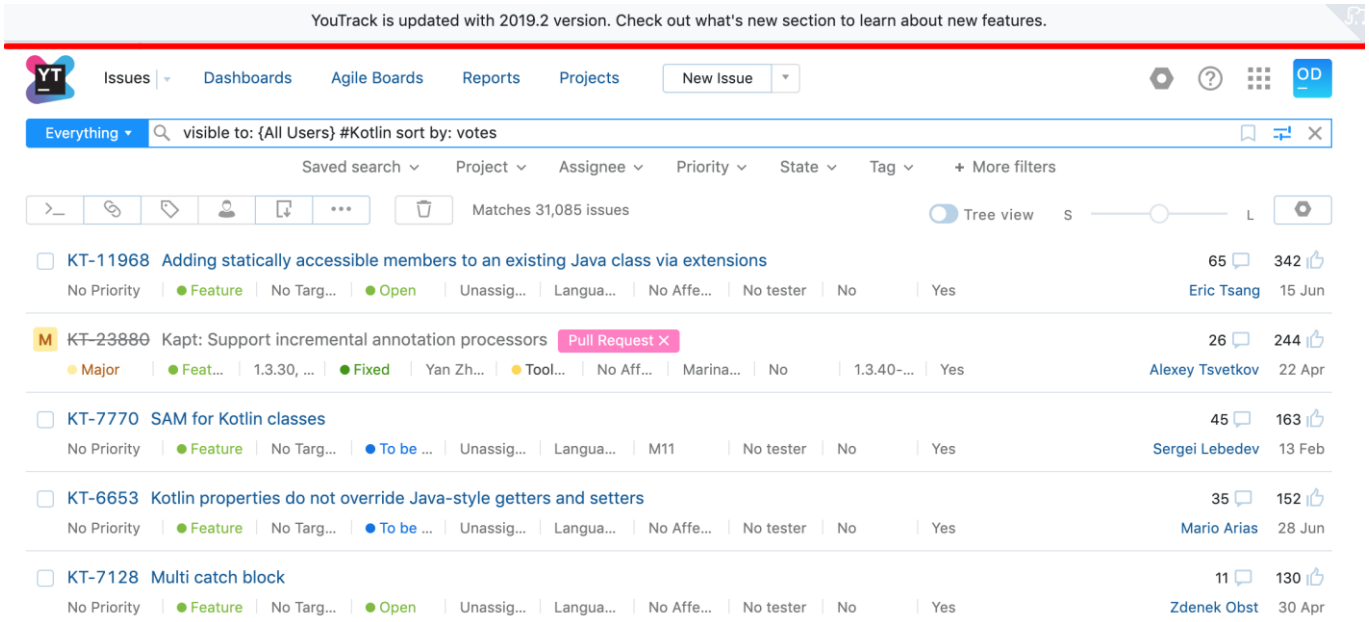


Рисунок 1.10 Інтерфейс сервісу YouTrack

Висновок до розділу 1

Всі системи менеджменту виробництва та відслідковування тікетів мають власних прихильників так, як мають власні особливості. Однак всі вони керуються людиною, що є спільною вразливістю чи недоліком. Явище, зване людським фактором, впливає на ефективність використання таких сервісів. Одним із проявів людського фактору може бути створення тікетів, які вже існують і над якими йде робота. Дублікати тікетів призводить до втрати робочого часу команд в цілому, тому є необхідність доповнення цих сервісів системою, яка запобігатиме, або попереджуватиме людський фактор. Так, на програмному рівні ITS відрізняються одна від одної, тому має сенс розробки окремого модуля, який дає оцінку подібності вмісту тікетів.

РОЗДІЛ 2. ТЕОРЕТИЧНА ЧАСТИНА. РОЗБІР АЛГОРИТМІВ ОЦІНКИ ПОДІБНОСТІ ТЕКСІВ

2.1. Схожість текстів.

Подібність тексту повинна визначати, наскільки "близькі" два фрагменти тексту як у поверхневій близькості (лексичної подібності), так і в значенні (семантичної подібності).

- На поверхні, якщо врахувати лише схожість на рівні слів, фрази можуть бути дуже схожими, якщо словниковий набір текстів близький. Зазвичай він не враховує фактичного значення слова або всієї фрази в контексті.
- Замість того, щоб робити порівняння слів до слів, нам також потрібно звернути увагу на контекст, щоб захопити більше семантики. Для розгляду смислової подібності нам потрібно зосередитись на рівнях фрази / абзацу (або лексичному рівні ланцюга), коли фрагмент тексту розбивається на відповідну групу споріднених слів перед обчисленням подібності.

2.2. Огляд способів обчислення схожості текстів.

Для розгляду взято такі способи обчислення схожості текстів:

- Відстань Жаккара
- Кластеризація методом к–середніх
- Косинус подібності
- Латентно-семантичний аналіз (Latent Semantic Indexing — LSI)
- Латентне розміщення Діріхле та дивергенція Дженсена-Шеннона
- Варіаційний автокодувальник

- Універсальний кодувальник речення
- Manhattan LSTM (MaLSTM)
- BERT + Косинус подібності

2.3. Відстань Жаккара

Відстань Жаккара – міра відмінності множин, є доповненням до коефіцієнта і визначається діленням різниці мір об'єднання і перетину двох множин на міру об'єднання.[1]

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

Приклад для двох речень:

Речення 1: AI is our friend and it has been friendly.

Речення 2: AI and humans have always been friendly.

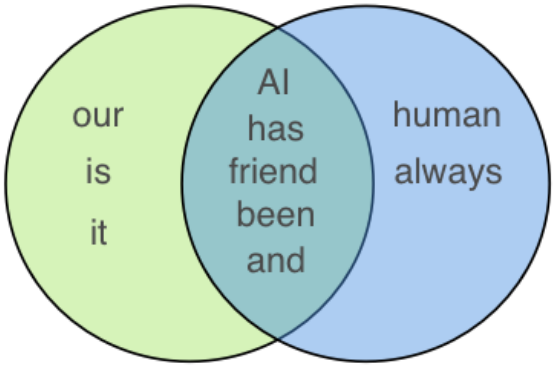


Рисунок 2.1 Спільні лексеми

Для того, щоб обчислити подібність, використовуючи подібність Жаккара, спочатку проведемо лематизацію, щоб звести слова до одного кореневого слова. У даному випадку “friend” та “friendly” обидва рахуються за “friend”, “has” та “have” аналогічно – “has”. Схожість Жаккара 5 / (5 + 3 +

2) = 0,5, що є величиною перетину множини, поділену на загальний розмір множини.

Інший приклад двох речень, але з подібним змістом:

Речення 1: President greets the press in Chicago

Речення 2: Trump speaks in Illinois

Дані два речення не мають спільних слів і матимуть оцінку Жаккара 0. Таке оцінювання дає той результат, на який розраховано, оскільки два речення мають дуже схоже значення. Тут подібність Жаккара не здатна зафіксувати ні семантичної, ні лексично-семантичної подібності цих двох речень.

Відстань Жаккара має очевидний недолік: в міру збільшення розміру тексту, кількість спільних слів, як правило, збільшується. Це призводить до збільшення і коефіцієнта, навіть якщо в порівнюваних текстах йдеться про різні теми.

2.4. Кластеризація методом к–середніх

Кластеризація методом к–середніх – це метод квантування векторів (перетворення якоїсь величини з неперервною шкалою значень на величину з дискретною шкалою значень) з метою розділення N елементів множини на «кластери» – підмножини, в яких містяться елементи, що найближчі до відповідного для кластера середнього значення k. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції $\sum_{i=1}^{N} d\left(x_i, m_j(x_i)\right)^2$, де d – метрика, x_i – і-тий елемент множини даних, а $m_j(x_i)$ – середнє значення кластера, якому на ітерації j приписаний елемент x_i.

Для використання методу к-середніх та похідних від нього потрібно спочатку створити вектори з речень. Кілька способів зробити перетворення це використовувати:

- **Множина слів за моделлю торба слів (bag-of-words model).** У цій моделі текст представляється у вигляді мультимножини його слів, не беручи до уваги граматику і навіть порядок слів, але зберігаючи множинність.
- **TF-IDF** (TF – term frequency, IDF – inverse document frequency).

TF (term frequency – частота слова) – відношення числа входжень обраного слова до загальної кількості слів документа. Таким чином, оцінюється важливість слова в межах обраного документа.

IDF (inverse document frequency – обернена частота документа) інверсія частоти, з якою слово зустрічається в документах колекції.

Використання IDF зменшує вагу широковживаних слів.

$$\mathrm{TF} = \frac{n_i}{\sum_k n_k}$$, де в чисельнику кількість входжень, а в знаменнику загальна кількість слів.

$$\mathrm{IDF} = \log \frac{|D|}{|(d_i \supset t_i)|}$$, де $|D|$ – кількість документів в колекції, $|\, d_i \supset t_i|$ – кількість документів, в яких зустрічається слово t_i коли $n_i \neq 0$. Вибір основи логарифму у формулі не має значення, адже зміна основи призведе до зміни ваги кожного слова на постійний множник, тобто вагове співвідношення залишиться незмінним.

Іншими словами, показник TF-IDF це добуток двох множників: TF та IDF. TF-IDF = TF * IDF [2]

- **Векторне представлення слів** з попередньо підготовлених методів, таких, як Fastext, Glove або Word2Vec, або

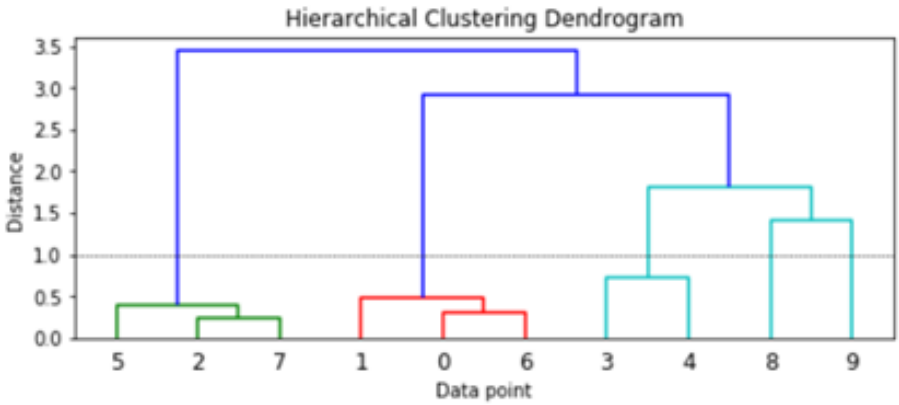
індивідуальних методів, використовуючи моделі Continuous Bag of Words (CBoW) або Skip Gram

Існує дві основні відмінності між BoW або TF-IDF у відповідності

- Bag of words або TF-IDF створюють одне число на кожне слово, тоді як векторне представлення слів зазвичай створює один вектор на кожне слово.
- Bag of words або TF-IDF хороші для класифікаційних документів в цілому, але векторне представлення слів добре для визначення контекстного вмісту.

Приклад застосування способів перетворення речень:

1. ‘The sky is blue and beautiful.’,
2. ‘Love this blue and beautiful sky!’,
3. ‘The quick brown fox jumps over the lazy dog.’,
4. “A king’s breakfast has sausages, ham, bacon, eggs, toast and beans”,
5. ‘I love green eggs, ham, sausages and bacon!’,
6. ‘The brown fox is quick, and the blue dog is lazy!’,
7. ‘The sky is very blue, and the sky is very beautiful today’,
8. ‘The dog is lazy, but the brown fox is quick!’,
9. ‘President greets the press in Chicago’,
10. ‘Obama speaks in Illinois’



	Document	Category	ClusterLabel
0	The sky is blue and beautiful.	weather	2
1	Love this blue and beautiful sky!	weather	2
2	The quick brown fox jumps over the lazy dog.	animals	1
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food	3
4	I love green eggs, ham, sausages and bacon!	food	3
5	The brown fox is quick and the blue dog is lazy!	animals	1
6	The sky is very blue and the sky is very beautiful today	weather	2
7	The dog is lazy but the brown fox is quick!	animals	1
8	President greets the press in Chicago	politics	4
9	Obama speaks in Illinois	politics	4

Рисунок 2.2 Ієрархічна кластеризація

Можна відзначити, що метод k-середніх дуже чутливий до масштабування і що в цьому випадку зважування IDF допомагає покращити якість кластеризації. Це поліпшення не помітно в коефіцієнті силуету, який є малим для обох, оскільки, ця міра страждає від явища, званого "концентрація вимірювання" – принцип, згідно з яким за певних досить загальних і не дуже обтяжливих обмежень значення функції великого числа змінних майже постійно. Для наборів даних високих розмірів, таких як текстові дані, рішення буде зменшувати розмірність векторів документів, застосовуючи прихований семантичний аналіз. Однак, щоб подолати проблему розмірності, існують такі заходи, як V-коефіцієнт Крамера та скоригований індекс Ранда, які є інформаційно-теоретичними оцінками,

оскільки вони базуються лише на призначенні кластерів, а не на відстані, отже, на них не впливає масштабування.

2.5. Косинус подібності

Косинус подібності означає подібність, вимірюючи косинус кута між двома ненульовими векторами. Таким чином, це судження про напрям, а не про модуль: два вектори з однаковим напрямом мають косинусну схожість рівну 1, два вектори, направлені на 90 ° відносно один одного, мають схожість 0, а два вектори, протилежно напрямлені, мають схожість -1, незалежно від їх модуля (довжини).

Перевага косинусу подібності полягає в тому, що два вектори, створені з двох документів, можуть бути на великій відстані, що означає різницю в розмірі документів, але важливим залишається кут між ними, косинус якого описує схожість двох документів.

2.6. Латентно-семантичне індексування (Latent Semantic Indexing — LSI)

Латентно-семантиичний аналіз (ЛСА) це техніка обробки природних мов, зокрема розподільної семантики, аналізу взаємозв'язків між набором документів і термінами, які вони містять, шляхом створення набору понять, пов'язаних з документами та термінами. Припускається, що слова, близькі за значенням, зустрічаються в подібних фрагментах матриці тексту, що містить кількість слів на документ (рядки представляють унікальні слова, а стовпці представляють кожен документ), побудована з великого фрагмента тексту. За допомогою математичного методу сингулярного розкладу матриці, зменшується кількість рядів матриці, зберігаючи структуру подібності у

стовпцях. Потім слова порівнюють за допомогою обчислення косинуса кута між двома векторами, що утворено будь-якими двома рядками. У контексті його застосування до пошуку інформації іноді його називають латентно-семантичним індексуванням (LSI).

Вважається, що основний смисловий простір колекції має меншу розмірність, ніж кількість унікальних лексем. Тому метод LSA застосовує аналіз основних компонентів на векторному просторі і зберігає лише напрямки у векторному просторі, які містять найбільшу дисперсію, тобто ті напрямки в просторі, які змінюються найшвидше, і, таким чином, передбачається, що містять більше інформації.

Припустимо, що обробляється три документи:

- d1 = “Shipment of gold damaged in a fire”
- d2 = “Delivery of silver arrived in a silver truck”
- d3 = “Shipment of gold arrived in a truck”

Метою роботи є знаходження найближчого по змісту документа при пошуку на запит : “gold silver truck”.

Встановлюються ваги лексем і ваги пошуку, що визначаються частотою знаходження лексеми. Створюється матриця лексем-документів A та матриця пошуку q:

Рисунок 2.3 Представлення речення та пошуку матрицями

Сингулярне представлення матриці A матиме вигляд $A = U \cdot S \cdot V^T$

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

Рисунок 2.4 Сингулярний розклад

Отримуємо матриці другого рангу, взявши перші два стовпці U і V і перші два стовпці та рядки S. В наслідок пониження рангу матриці зменшується проблема виявлення синонімів, об’єднанням вимірів пов’язаних слів. Також частково зменшується проблема багатозначності.

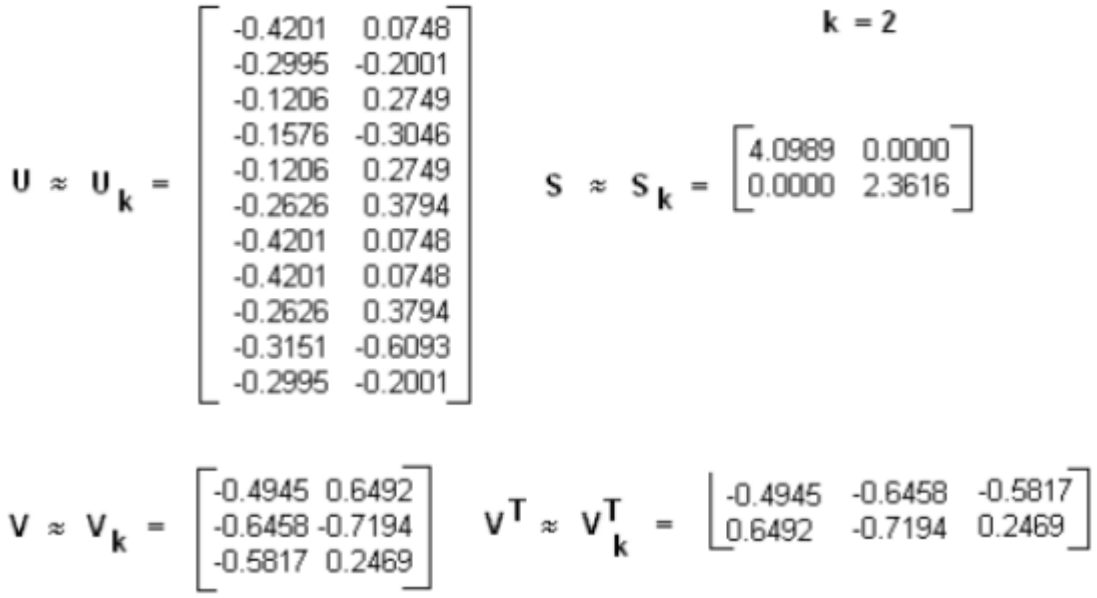


Рисунок 2.5 Матриці другого порядку

Знаходимо нові координати вектора документа в цьому зменшеному двовимірному просторі. Рядки матриці V містять значення власного вектора. Це координати окремих векторів документів, отже:

- d1 = (-0.4945, 0.6492)
- d2 = (-0.6458, -0.7194)
- d3 = (-0.5817, 0.2469)

Знаходимо нові координати вектору запиту у зменшеному двовимірному просторі.

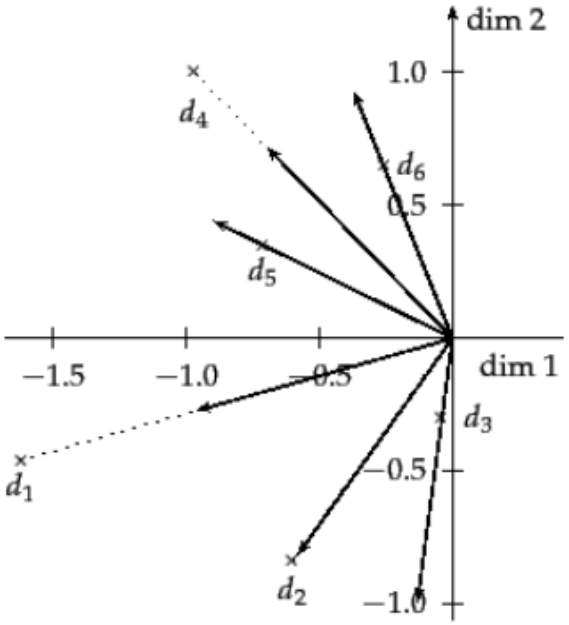


Рисунок 2.6 Власні вектори на координатній площині

Це нові координати вектору пошуку у двох вимірах. Матриця q тепер відрізняється від початкової:

$$q = q^T U_k S_k^{-1} \qquad k = 2$$

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$q = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

Рисунок 2.7 Власний вектор рядка пошуку

Тепер застосовується косинус подібності до матриці пошуку q та речень d_1, d_2, d_3.

$$\mathbf{sim(q,\ d) = \frac{q \bullet d}{|q|\,|d|}}$$

$$\mathbf{sim(q,\ d_1)} = \frac{(-0.2140)\,(-0.4945) + (-0.1821)\,(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

$$\mathbf{sim(q,\ d_2)} = \frac{(-0.2140)\,(-0.6458) + (-0.1821)\,(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$\mathbf{sim(q,\ d_3)} = \frac{(-0.2140)\,(-0.5817) + (-0.1821)\,(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$\mathbf{d_2 > d_3 > d_1}$$

Рисунок 2.8 Обчислення косинусу подібності

За косинусом подібності оцінка схожості d_2 до запиту найбільша серед інших.

2.7. Латентне розміщення Діріхле та дивергенція Дженсена-Шеннона

В обробці природної мови прихований розподіл Діріхле (LDA - latent Dirichlet allocation) – це генеративна статистична модель, що дозволяє пояснювати результати спостережень за допомогою неявних груп, завдяки чому можливе виявлення причин подібності деяких частин даних. Наприклад, якщо спостереженнями є слова, зібрані в документи, стверджується, що кожен документ являє собою суміш невеликої кількості тем і що поява кожного слова пов'язане з однією з тем документа. LDA є одним з методів тематичного моделювання.

LDA має безліч застосувань:

- Розрізнення тем, що зустрічаються в документі
- Отримання кращого розуміння типів множини документів (будь то про новини, статті у вікіпедії, ділові документи)
- Кількісне визначення найбільш вживаних / найважливіших слів у множині
- Порівняння схожості документів

На високому рівні модель передбачає, що кожен документ буде містити кілька тем, так що в документі є перекриття тем. Слова в текстах належать темам, які в свою чергу невідомі і, більш того, їх і не потрібно вказувати, проте кількість тем має бути визначена. Нарешті, між темами можуть бути слова, що перетинаються, тому декілька тем можуть мати спільні слова.

Модель генерує дві приховані змінні:

1. Розподіл тем за кожним документом.
2. Розподіл слів по кожній темі.

Після навчання кожен документ матиме дискретний розподіл загальних тем, і кожна тема матиме дискретний розподіл по всіх словах. Маємо розподіл тем для нового документа.

Мета – знайти найбільш схожі документи в множині. Для цього порівнюється тематичний розподіл нового документа з усіма тематичними розподілами документів у множині. Метрика відстані Дженсена-Шеннона використовується, щоб знайти найбільш подібні документи. Метод Дженсена-Шеннона це метод вимірювання подібності між двома розподілами ймовірностей, також відомий як радіус інформації. Дженсен-Шеннон симетричний, на відміну від Кульбака-Лейблера, на якому заснована

формула. Це добре, тому що необхідно, щоб подібність між документами A і B була такою ж, як і подібність між B і A.

Для дискретних розподілів P і Q, розбіжність Дженсена-Шеннона, *JSD* визначається як:

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M),$$

де $M = \frac{1}{2}(P + Q)$ та D це розбіжність Кульбака-Лейблера:

$$D(P||Q) = \sum_i P(i)\log\left(\frac{P(i)}{Q(i)}\right)$$

Чим менше відстань Дженсена-Шеннона, тим більше подібних двох розподілів (і в нашому випадку тим більше подібних будь-яких 2 документів).

Створення середнього документа M між цими двома документами 1 і 2 відбувається шляхом усереднення їх розподілу ймовірностей або об'єднаннями вмісту обох документів. Вимірювання різниці кожного з документів 1 та 2 з середнім документом M, а потім порівняння цих різниць показує міру схожості документів 1 та 2.

2.8. Варіаційний автокодувальник

Задача моделі варіаційного автокодувальника це передбачення вхідної інформації, конкретніше розглянемо штучну нейронну мережу автокодувальника. Вона намагається навчитися наближати функцію ідентичності

$$f_{w,b}(x) \approx x$$

Намагаючись навчити мережу, що спочатку може здатися тривіальним, важливо зазначити, що ми хочемо навчитися стислому представленню даних, таким чином знаходячи структуру. Це можна зробити, обмеживши кількість

прихованих одиниць у моделі. Такі види автокодувальників називаються неповними.

Зображення того, що може навчитися автокодувальник:

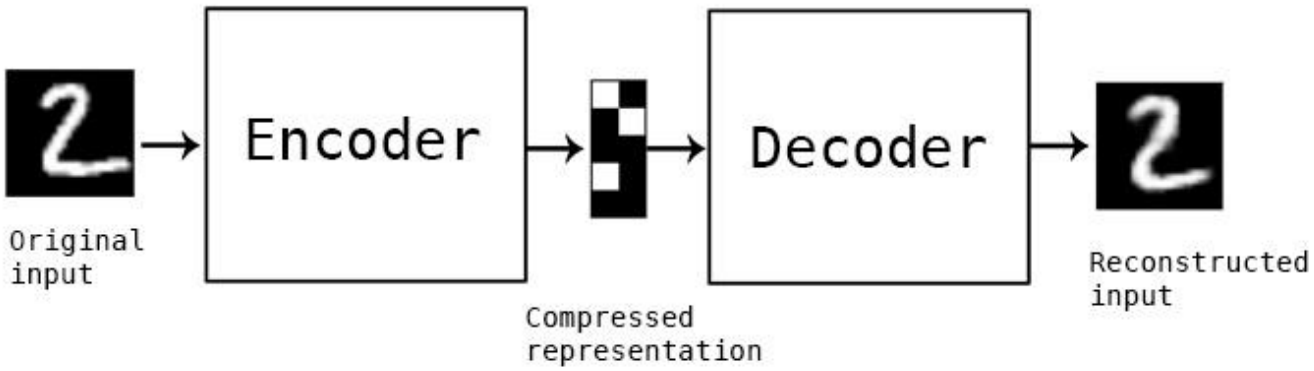


Рисунок 2.9 Приклад навченого автокодувальника

Ключовою проблемою буде отримання проекції даних в одному вимірі без втрати інформації. Коли цей тип даних проектується в латентному просторі, багато інформації втрачається, і перетворити її до початкової форми майже неможливо. Незалежно від того, скільки застосованих операцій, оригінальні дані не можуть бути відновлені.

Вирішення проблеми полягає у тому, що нейронні мережі мають властивість згинати простір, щоб отримати лінійні данні. Архітектури автокодування застосовують цю властивість у своїх прихованих шарах, що дозволяє їм навчатися уявлення низького рівня у прихованому просторі подання.

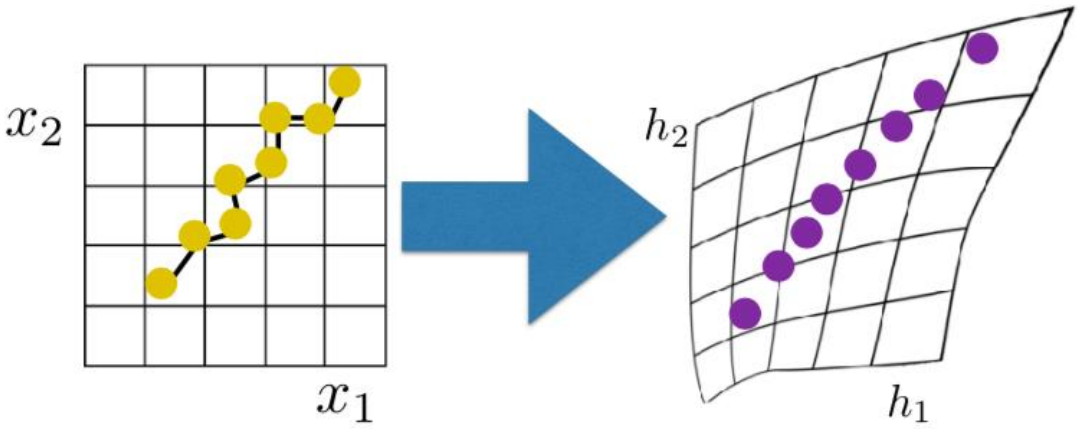


Рисунок 2.10 Згинання простору

Типова архітектура автокодування складається з трьох основних компонентів:

- Архітектура кодування, що складається з ряду шарів зі зменшенням кількості вузлів і в кінцевому рахунку зводиться до прихованої репрезентації виду.
- Прихована репрезентація виду, яка відображає простір найнижчого рівня, в якому зменшується введення і зберігається інформація.
- Архітектура декодування. Дзеркальне зображення архітектури кодування, але в якому кількість вузлів у кожному шарі збільшується і в кінцевому підсумку виводиться аналогічний ввід.

Дані закодовуються до прихованих (випадкових) змінних, а потім приховані змінні розкодовуються для реконструкції даних.

Замість того, щоб безпосередньо виводити значення для латентного стану, як це було б у стандартному автокодувальнику, модель варіаційного автокодувальника виводить параметри, що описують розподіл для кожного виміру в латентному просторі. Оскільки припускається, що використовується нормальний розподіл, ми виведемо два вектори, що описують середнє значення та дисперсію латентних розподілів стану.

Якби ми побудували справжню багатоваріантну модель Гауса, нам потрібно було б визначити коваріаційну матрицю, що описує кореляцію кожного з вимірів. Однак ми зробимо спрощене припущення, що наша матриця коваріації має лише ненульові значення по діагоналі, що дозволяє описати цю інформацію в простому векторі.

Потім модель декодувальника генерує прихований вектор шляхом вибірки з цих визначених роз

У звичайних детермінованих автокодеках приховані данні не навчаються розподілу ймовірностей даних, отже, він не підходить для генерування нових даних. поділів і продовжить розробку реконструкції вихідного вводу.

Модель варіаційного автодокувальника вирішує проблему так, як чітко визначає розподіл ймовірності на прихованих змінних і засвоює приховані подання входів не як одиничні точки, а як м'які еліпсоїдальні області в латентному просторі, змушуючи приховані подання заповнювати прихований простір, а не запам'ятовувати входи як пунктуальні, ізольовані приховані подання.

Використовуючи модель варіаційного автодокувальника, ми можемо підігнати параметричний розподіл (у даному випадку Гауса). Саме це відрізняє варіаційного автодокувальника від звичайного автокодувальника, який покладається лише на вартість реконструкції. Це означає, що під час запуску, коли ми хочемо витягувати зразки з мережі, все, що потрібно зробити, - це генерувати випадкові вибірки з нормального розподілу і подавати їх в кодувальник, який буде генерувати вибірки.

Використовувати варіаційний автокодувальник можна і для навчання прихованих шарів текстовими даними, матрицями слів, зіставлення даних у прихованих змінних, навчати модель відрізняти різні теми документи.

2.9. Універсальний кодувальник речення

Універсальний кодувальник речення це модель, що закодовує текст у багатовимірні вектори, які можна використовувати для класифікації тексту, розпізнавання семантичної подібності, кластеризації та інших завдань обробки природних мов. Вона підготовлена та оптимізована для більшого за одне слово тексту: речень, фраз, коротких параграфів. На ввід до моделі подається текст будь-якої довжини, а на виході отримується вектор 512-розмірний вектор. Цю модель застосовують до STS benchmark для знаходження схожості документів. Універсальний кодувальник речення навчається на deep averaging network кодувальниках. Декілька таких кодувальників це:

- InferSent – розробка фейсбуку, BiLSTM натренована на SNLI наборі даних, 570 тисяч пар англійських речень позначені однією з трьох категорій: невизначене значення, суперечливість або нейтральність (entailment, contradiction or neutral).
- Google Sentence Encoder: простіша мережа глибокого усереднення (DAN).

2.10. Manhattan LSTM (MaLSTM)

Сіамські нейромережі – це мережі, які мають у собі дві або більше однакових підмереж. Вони показують дуже хороші результати у задачах визначення подібності і використовуються у визначенні семантичної подібності документів, розпізнавання зображень, тощо.

Сіамська – це назва загальної архітектури моделі, де модель складається з двох однакових підмереж, які обчислюють певний вид представлення векторів для двох входів, а вимірювання відстані використовується для оцінки схожості або різниці входів.

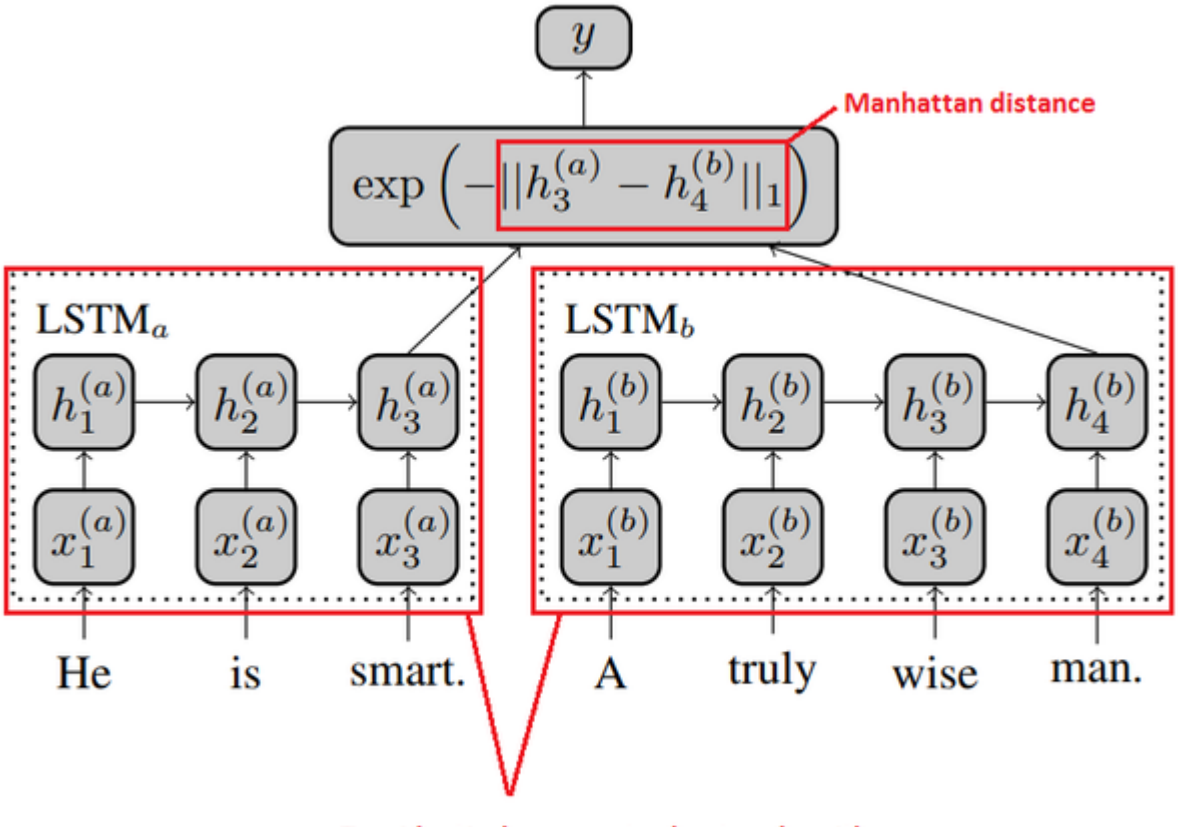


Рисунок 2.11 Сіамські нейромережі та відстань Мангеттена

MaLSTM (Manhattan LSTM) посилається на той факт, що використовується відстань Мангеттена для порівняння кінцевих прихованих станів двох стандартних шарів LSTM. Такі альтернативи, як косинус або евклідова відстань, також можуть бути використані, але прийнято, що Манхеттенська відстань перевершує інші розумні альтернативи, такі як схожість косинусу.

Данні проектуються в простір, в якому схожі елементи стискаються, а різні розподіляються по вивченому простору. Це дуже ефективно, адже мережі обмінюються параметрами.

2.11. Bidirectional Encoder Representations from Transformers (BERT)

Інструменти мовного моделювання, такі як ELMO, GPT-2 та BERT, дозволяють отримати вектори слів, які перетворюють знання їх місця та оточення.

Мета – показати, що вектори слова BERT перетворюються на контекст. Візьмемо для прикладу наступні три речення:

- record the **play**
- **play** the record
- **play** the game

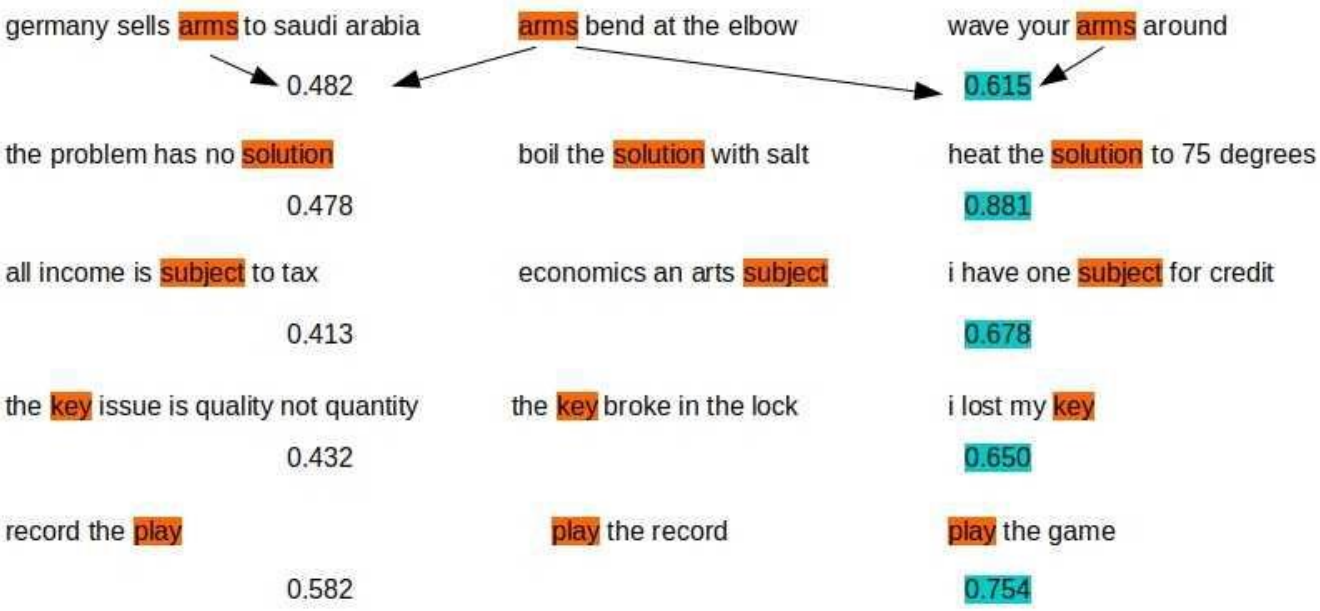


Рисунок 2.12 Оцінка подібності значення слів

Значення слова “play” у другому реченні має бути більш схоже до такого ж слова, використане у третьому реченні та менш схоже до того, що у першому реченні. Можна використовувати будь-які трійки, такі, як наведено вище, в будь-якій кількості для перевірки точності роботи BERT. Ось декілька таких трійок, і результати показують, що заздалегідь навчений BERT здатний з'ясувати контекст, у якому це слово використовується:

Кожен рядок показує три речення. Речення в середині виражає той же контекст, що і речення праворуч, але відрізняється від того, що знаходиться зліва. Усі три речення у рядку мають спільне слово. Числа показують обчислений косинус подібності між зазначеними парами слів. Після розпізнавання за допомогою BERT видно, що для слова в середині оцінка схожості більша для слова праворуч, ніж для того, що ліворуч.

Висновок до розділу 2

В розділі було описано декілька методів для аналізу текстів. Всі ці методи в певній мірі мають переваги один над іншим в залежності від мети з якою їх застосовують. Алгоритми на основі нейронних мереж можуть класифікувати тексти за темами, а тому придатні розрізняти саме про що йдеться в тексті в тій чи іншій мірі. Алгоритми без застосування нейронних мереж оперують більш примітивними поняттями як розташування та набори слів, а отже можуть використовуватись для визначення змін у тексті і оцінювати на скільки текст відрізняється від своєї попередньої версії.

РОЗДІЛ 3. ОПИС ІНСТРУМЕНТІВ РОЗРОБКИ

Однією із вимог до системи захисту від створення дублікатів задач є її самостійність. Вона має представляти собою модуль, або плагін для того, щоб її використання було можливе у різних системах менеджменту тікетів. Отже розроблювана система повинна надавати інтерфейс, який можна використовувати і таким чином робить її самостійною навіть без інтеграції до іншого сервісу.

На основі веб-фреймворку Django на мові програмування Python можна створювати самостійні модулі веб-сервісу, які можна переносити до інших проектів чи сервісів без зміни.

3.1. Мова програмування Python

Python – це інтерпретована, високорівнева мова програмування загального призначення. Філософією цієї мови підкреслює важливість читабельності коду водночас надаючи можливість написати чіткий, логічний код для проектів різного масштабу.

Python динамічно типізований та має комбінацію індексування посилань та збирач сміття для автоматизації розподілення використання оперативної пам’яті комп’ютера. Мова підтримує декілька парадигм програмування: структурне, частково процедурне, об’єктно-орієнтоване, функціональне програмування.

Мова Python є безкоштовною з відкритим кодом, так званий open-source. Стандартний дистрибутив має безліч модулів, включаючи модуль для розробки GUI. Станом на листопад 2019 репозиторій сторонніх бібліотек Python – PyPi налічує більше 200 тисяч бібліотек, призначених для автоматизації, аналізу даних, роботою з базами даних, машинним навчанням, тощо.

Програми Python часто використовують пакети та модулі, які не входять до стандартної бібліотеки. До того ж деякі програми працюють тільки з конкретними версіями сторонніх модулів. Для того, щоб програми мали ізольовані версії модулів існують віртуальні середовища. Так, кожна програма матиме власне ізольоване від інших середовище і копію програми pip, яка і встановлює модуль з PyPi у середовище, в якому знаходиться.

Також програми написані на Python переносні, що власне більшості інтерпретованих мов. і працює майже на всіх відомих платформах — від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android.

Інтерпретатор цієї мови може бути розширений додатками, розробленими на C/C++ або іншою мовою, яку можна викликати з C. Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження. Існує спеціальна версія Python для віртуальної машини Java — Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, яка підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на ньому. Також кілька проектів забезпечують інтеграцію з платформою Microsoft.NET.

3.2. Django

Django це безкоштовний веб-фреймворк на Python з відкритим кодом. Архітектурний патерн фреймворка MTV (model-template-view). Основна мета Django - полегшити створення складних веб-сайтів, керованих базами даних. Фреймворк наголошує на повторному використанні та підключеності компонентів, меншій кількості коду, швидкому розвитку та принципі не

повторювати себе. Python використовується на всьому протязі, навіть для файлів налаштувань та моделей даних. Django також надає додатковий адміністративний інтерфейс для створення, читання, оновлення та видалення, який динамічно генерується за допомогою самоаналізу та налаштовується через адміністраторські моделі.

Незважаючи на наявність власної номенклатури іменування об'єктів, що викликаються, генеруючи відповіді HTTP, “view”, основна структура Django може розглядатися як архітектура MVC. Фреймворк складається з ORM – object-relational map, яка опосередковує між моделями даних (визначеними класами Python) та реляційною базою даних ("Molel"), системою обробки HTTP-запитів із системою веб-шаблонів ("View") та диспетчер URL-адрес на основі регулярних виразів ("Controller").

До основної бази також входять:

- легкий і окремий веб-сервер для розробки та тестування
- система серіалізації та перевірки форм, яка може переводити між формами HTML і значеннями, придатними для зберігання в базі даних
- шаблонна система, яка використовує поняття наслідування, запозичене з об'єктно-орієнтованого програмування
- кешувальний фреймворк, який може використовувати будь-який з декількох способів кешування
- підтримка класів проміжного програмного забезпечення, які можуть втручатися на різних етапах обробки запитів та виконувати спеціальні функції
- внутрішня система диспетчера, яка дозволяє компонентам програми передавати події один одному заздалегідь визначеними сигналами
- система інтернаціоналізації, що включає переклади власних компонентів Django на різні мови

- система сериализації, яка може записувати і читати XML, JSON в екземпляри моделі Django
- система для розширення можливостей шаблонного двигуна
- інтерфейс до unit-test модуля Python
- Фреймворк Django REST - це потужний і гнучкий інструментарій для створення веб-API.

Основний дистрибутив Django також поєднує ряд програм у своєму пакеті "contrib", включаючи:

- розширювана система аутентифікації
- динамічний адміністративний інтерфейс
- інструменти для створення RSS і Atom каналів синдикації
- Фреймворк "Sites", який дозволяє одній установці Django запускати кілька веб-сайтів, кожен з яких має власний вміст та програми
- інструменти для створення Google Sitemap
- вбудоване в пом'якшенні для міжсайтового підроблення запиту, міжсайтового скриптинга , ін'єкцій SQL , пароль розтріскування і інших типових веб - атак, більшість з них включено за замовчуванням
- фреймворк для створення GIS- додатків

Система конфігурації Django дозволяє вбудовувати сторонній код до звичайного проекту за умови, що він дотримується конвенцій програми багаторазового використання. Більше 2500 пакетів доступні для розширення оригінальної поведінки фореймворку, забезпечуючи вирішення проблем, з якими не вирішувався оригінальний інструмент: реєстрація, пошук, надання та споживання API , CMS тощо.

Однак ця розширюваність пом'якшується залежностями внутрішніх компонентів. Хоча філософія Django передбачає слабкі залежності, фільтри

шаблонів і теги передбачають одну реалізацію двигуна, і як пакети програм Auth, так і адміністратори вимагають використання внутрішньої ORM. Жоден з цих фільтрів або пакетних програм не є обов'язковим для запуску проекту Django, але багаторазові програми, як правило, залежать від них, спонукаючи розробників продовжувати використовувати офіційний стек, щоб повною мірою скористатися екосистемою додатків.

Django офіційно підтримує чотири бази даних: PostgreSQL, MySQL, SQLite та Oracle. Microsoft SQL Server може використовуватися з django-mssql в операційних системах Microsoft, тоді як зовнішні пакети даних існують і для IBM Db2, SQL Anywhere та Firebird. Є модуль з назвою django-nonrel, який підтримує бази даних NoSQL, такі як MongoDB та Datastore Google App Engine.

3.3. SQLite [9]

SQLite – система управління реляційними базами даних (RDBMS), що міститься в бібліотеці C. На відміну від багатьох інших систем управління базами даних, SQLite не є двигуном бази даних клієнт-сервер. Скоріше, вона вбудована в кінцеву програму.

SQLite сумісний з ACID і реалізує більшість стандартів SQL, як правило, з наслідуванням синтаксису синтаксису PostgreSQL. Однак SQLite використовує динамічний і слабо набраний синтаксис SQL, який не гарантує цілісність домену. Це означає, що можна, наприклад, вставити рядок у стовпчик, визначений як ціле число. SQLite намагатиметься перетворити дані між форматами, де це доцільно, рядок "123" в ціле число в цьому випадку, але не гарантує таких перетворень, і збереже дані як такі, як таке перетворення неможливе.

SQLite є популярним вибором, як вбудоване програмне забезпечення бази даних для локального / клієнтського зберігання в прикладному програмному забезпеченні, такому як веб-браузери. Це, мабуть, найбільш широко розгорнутий двигун бази даних, оскільки його сьогодні використовують декілька широко розповсюджених браузерів, операційних систем та вбудованих систем. SQLite має прив'язку до багатьох мов програмування.

На відміну від систем управління базами даних клієнт-сервер, SQLite не має автономних процесів, з якими прикладна програма спілкується. Натомість бібліотека SQLite пов'язана між собою і, таким чином, стає невід'ємною частиною прикладної програми. Зв'язок може бути статичним або динамічним. Прикладна програма використовує функціональність SQLite через прості виклики функцій, які зменшують затримку в доступі до бази даних: виклики функцій в рамках одного процесу є більш ефективними, ніж міжпроцесовий зв'язок. SQLite зберігає всю базу даних (визначення, таблиці, індекси та самі дані) у вигляді єдиного крос-платформного файлу на хост-машині. Він реалізує цю просту конструкцію, блокуючи весь файл бази даних під час написання. Операції з читання SQLite можуть бути багатозадачними, хоча запис можна виконувати лише послідовно.

Через дизайн без сервера, додатки SQLite потребують меншої конфігурації, ніж бази даних клієнт-сервер. SQLite називається zero-conf, оскільки він не потребує управління сервісом або управління доступом на основі GRANT та паролів. Контроль доступу обробляється за допомогою дозволів файлової системи, наданих самим файлам бази даних. Бази даних в системах клієнт-сервер використовують права файлової системи, які надають доступ до файлів бази даних.

SQLite використовує PostgreSQL як референтну платформу. Одне головне відхилення полягає в тому, що, за винятком первинних ключів, SQLite не застосовує перевірку типу; тип значення є динамічним і не є строго

обмеженим схемою (хоча схема спричинить перетворення під час зберігання, якщо таке перетворення потенційно є оборотним).

Ще одне значення безсерверного дизайну полягає в тому, що кілька процесів можуть не мати змоги записати у файл бази даних. У серверних базах даних кілька авторів підключаються до одного образу, який здатний внутрішньо обробляти його блокування. З іншого боку, SQLite має покладатися на блокування файлової системи. Він має менше знань про інші процеси, які одночасно отримують доступ до бази даних. Отже, SQLite не є кращим вибором для інтенсивних розгортань. Однак для простих запитів з невеликою паралельністю продуктивність SQLite отримує перевагу від уникнення накладних витрат на передачу своїх даних іншому процесу.

Висновок до розділу 3

У цьому розділі було розглянуто використані інструменти підчас розробки дипломного проекту. Мова програмування Python дозволяє розширювати вже існуючі системи, навіть написані на іншій мові, а тому підходить для розробки додатків під всі системи. В додаток до особливості мови програмування Python, Django, як веб-фреймворк дозволяє розробляти самостійні модулі, які можна підключати до інших проектів без зайвої конфігурації та змін у коді.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

В процесі розробки було створено три застосунки Django:

1. Стандартний застосунок, який існує у всіх сервісах Django. В ньому містяться конфігурації проекту: налаштування підключення до баз даних, конфігурації роутингу проекту, Web Server Gateway Interface конфігурації, тощо. Застосунок називається Application.
2. Застосунок, в якому описано демонстраційний сервіс менеджменту – ITS система. Застосунок називається IssueTracking. Цей застосунок має реалізовані основні функції ITS: можливість створювати issue, або тікети, редагувати їх, видаляти за наявності прав адміна, отримувати всі задачі у вигляді списку карток.
3. Останній модуль – це компонент, який, як і всі інші можна брати з проекту і переносити його в інший. У цій компоненті DuplicationDetection описані самостійні маршрути, запити та шаблон HTML файлу, який Django дозволяє наслідувати з іншого, базового шаблону, який існує в іншому проекті.

В директорії проекту є дві субдиректорії: src та venv.

Venv це ізольоване середовище, в якому встановлені дистрибутиви PyPi для проекту. Командою термінала pip freeze отримуємо список модулів:

asgiref==3.2.7

astroid==2.4.1

autopep8==1.5.2

colorama==0.4.3

Django==3.0.6

django-crispy-forms==1.9.1

isort==4.3.21

lazy-object-proxy==1.4.3

mccabe==0.6.1

pycodestyle==2.6.0

pylint==2.5.2

pytz==2020.1

six==1.14.0

sqlparse==0.3.1

toml==0.10.1

typed-ast==1.4.1

wrapt==1.12.1

При установці модуля Django необхідні йому інші модулі автоматично установлюються з ним.

Директорія src наповнюється кодом проекту.

django-admin startproject команда терміналу створює перший компонент, який описує файли конфігурації.

Командами **python manage.py startapp IssueTracking** та

python manage.py startapp DuplicationDetection

створюються два останні компонента.

В кінці розробки директорія виглядає таким чином:

```
SRC
│   db.sqlite3
│   manage.py
│
├────Application
│       asgi.py
│       settings.py
│       urls.py
│       wsgi.py
│       __init__.py
│
├──── DuplicationDetection
│   │   admin.py
│   │   apps.py
```

```
│   │   models.py
│   │   tests.py
│   │   urls.py
│   │   views.py
│   │   __init__.py
│   │
│   ├───middleware
│   │       difference_calculator.py
│   │
│   ├───migrations
│   │       __init__.py
│   │
│   ├───templates
│           submition_prestriction_view.html
│
├───IssueTracking
│   │   admin.py
│   │   apps.py
│   │   forms.py
│   │   models.py
│   │   tests.py
│   │   urls.py
│   │   views.py
│   │   __init__.py
│   │
│   ├───migrations
│   │   │   0001_initial.py
│   │   │   __init__.py
│   │
```

```
│   ├──templates
│   │       index.html
│   │       issue_add.html
│   │       issue_detail.html
│   │       issue_update.html
│   │
│
└──templates
        base.html
        sidebar.html
```

4.1. Компонент IssueTracking

Django генерує таблиці в базі даних на основі DAO, описаних в models.py та файлах модулів установлених у віртуальному середовищі.

Клас IssueTracking.moleds.Issue це DAO, в якому описані значення таблиці:

- заголовок
- опис
- дату створення
- користувача, що створив тікет. Це залежне поле від іншої таблиці Users, створеної стандартною компонентою Django.
- значення, яке прийнятне для адресної строки браузера та унікальне для кожного з тікетів

Клас також містить методи, які необхідні для Django, щоб обробляти логіку записування до бази даних та запитів з неї.

У папці templates містяться .html файли із застосуванням шаблонів Django.

Таблиця 4.1 структура компоненти IssueTracking

Назва файлу	**Функції, які реалізує**
IssueTracking.admin.py	Реєструє DAO тікетів в системі адміністратора
IssueTracking.forms.py	Описує форму на основі класів Django. Форма відрисовується в браузері і заповнюється підчас створення і редагування тікетів.
IssueTracking.moleds.py	Описує DAO тікета
IssueTracking.tests.py	Містить unit тести до компонента
IssueTracking.urls.py	Описує маршрути компоненти. Ці файли в кожному модулі напряму зв’язані з головним модулем і реєструє маршрути.
IssueTracking.views.py	Описує обробку запитів на посилання на основі класів Django. Описані класи імпортовані до файлу urls.py модуля IssueTracking і прив’язують запити на посилання до логіки їх обробки

4.2. Компонент DuplicationDetection

У цій компоненті не описано DAO, адже використовується той об’єкт, який імпортовано в компонент. Тому і в файлі admin.py нічого не описано, але ці файли є обов’язковими для кожного компонента Django.

Папка templates містить .html файл, який можна наслідувати від базових шаблонів проекту, в який підключено цей компонент.

Файл views.py містить один клас обробки запитів на основі класів Django. Ця логіка обробки запитів використовується як прошарок і адресу переадресації з основного компонента IssueTracking, що зменшує кількість налаштувань та змін самого коду при підключенні модуля.

Папка middleware містить інтерфейс, який реалізується алгоритмами оцінки подібності текстів. Такий підхід робить алгоритми взаємозамінними, тобто достатньо лише створити нову імплементацію інтерфейсу.

4.3. Інструкція ористувача

Користування сервісом починається з авторизації користувача логіном та паролем, який був створений адміністратором. Після успішної авторизації домашня сторінка сервісу стає доступною.

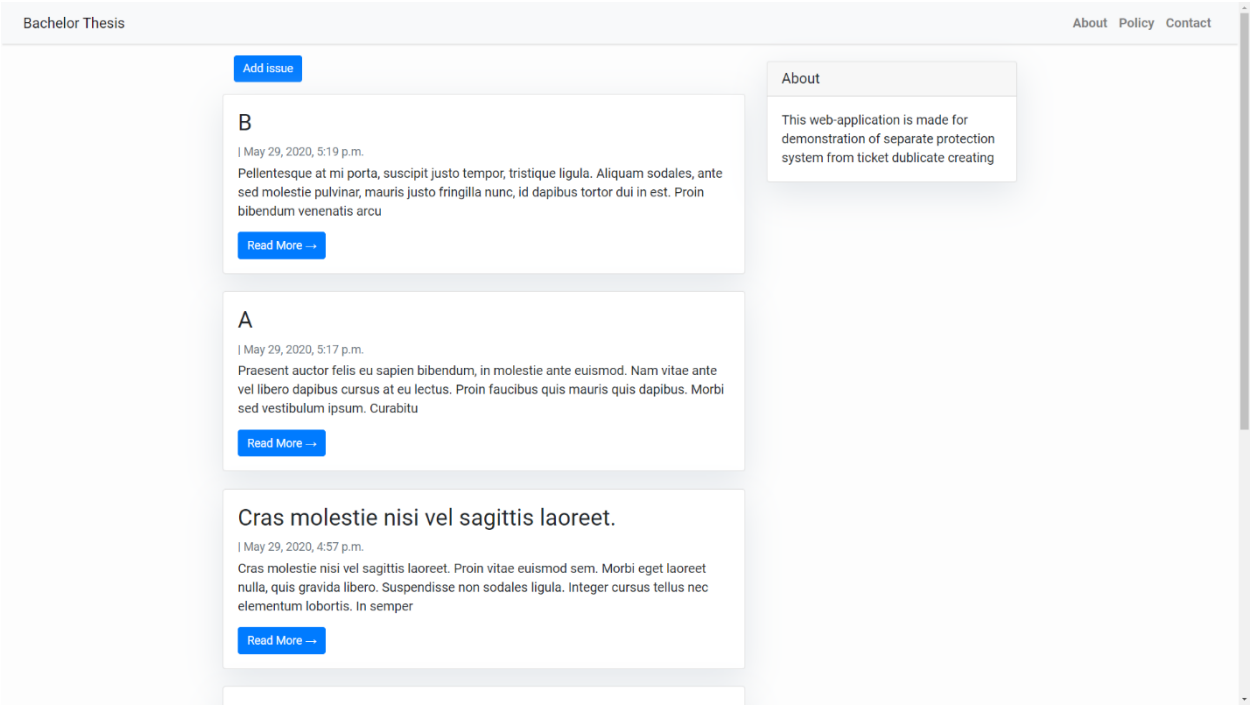


Рисунок 4.1. Домашня сторінка

На домашній сторінці списком карток показано існуючі тікети. На картках вказаний заголовок, дата створення та частина опису із тікету.

Для перегляду повного опису потрібно натиснути кнопку на картці “Read More →”, відкриється повний вміст тікету.

У відкритому тікеті знаходиться повна інформація: заголовок, ім’я того, хто створив тікет, дата створення та кнопка, що дозволяє редагувати відкритий тікет.

Редагування тікету має вигляд форми, яка заповнена попередньо створеним тікетом. Змінити можна лише заголовок, опис та стан виконання задачі. Змінити автора, дату створення неможна.

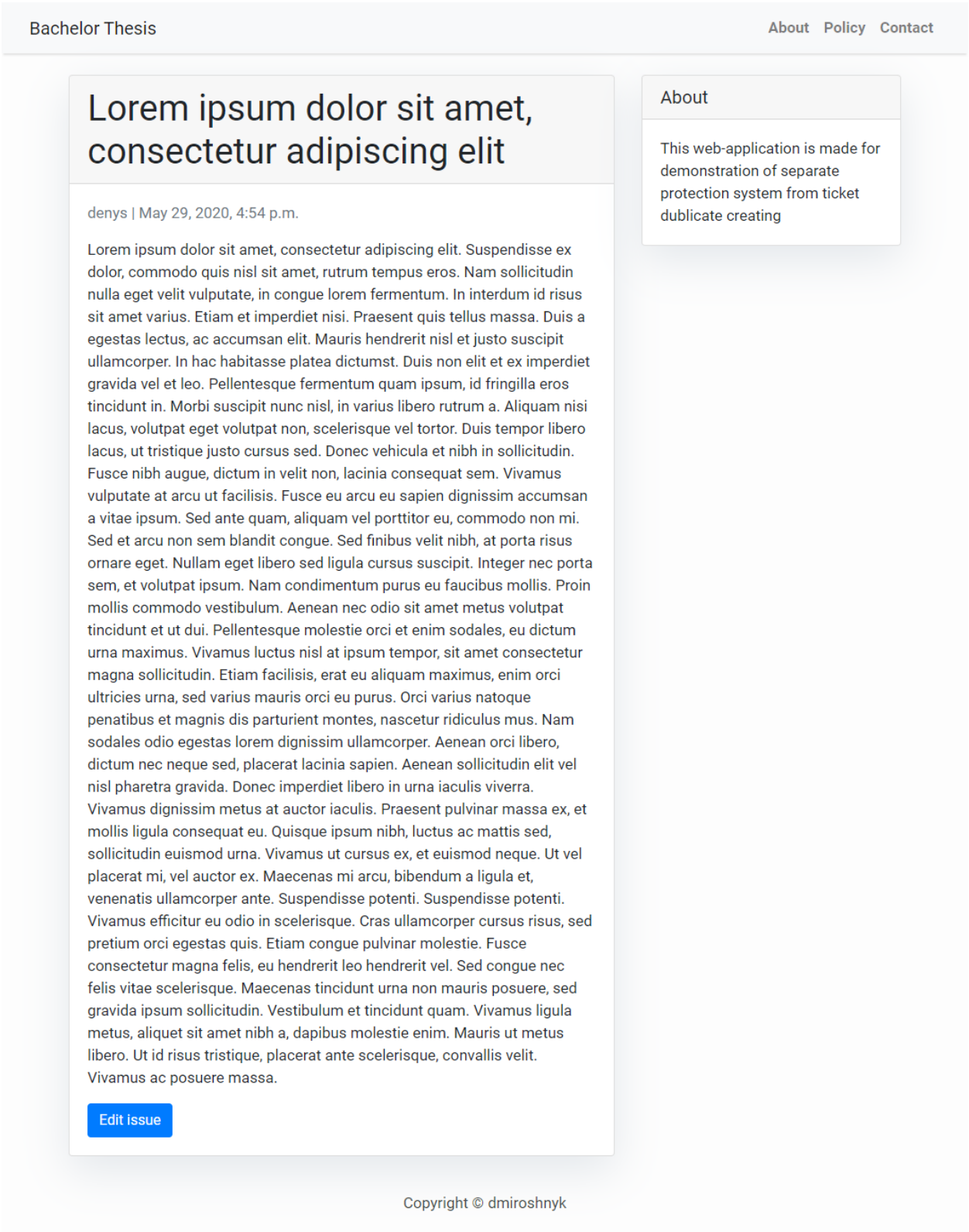


Рисунок 4.2. Відкритий тікет

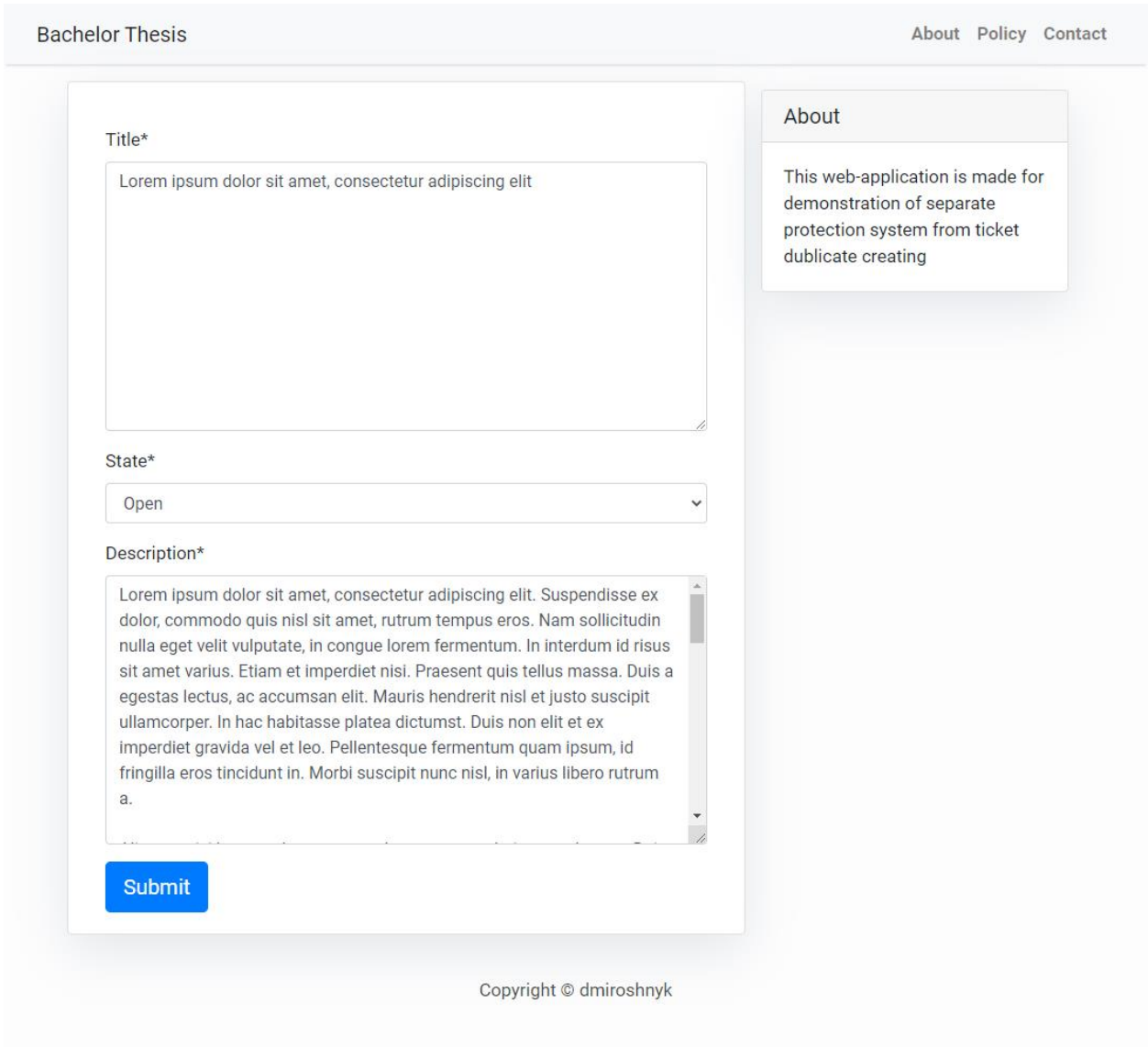


Рисунок 4.3 Тікет відкритий на редагування.

Щоб створити новий тікет потрібно натиснути на кнопку “Add issue” на домашній сторінці. Натиснувши, ви перейдете до пустої форми. Вона схожа на форму редагування. Дата створення і автор тікета заповнюється автоматично і не може бути змінений. Поле з унікальним, прийнятним для адресної строки значенням, створюється на основі заголовка, тому символи, які можна використовувати у заголовку – обмежені до латиниці та деяких символів.

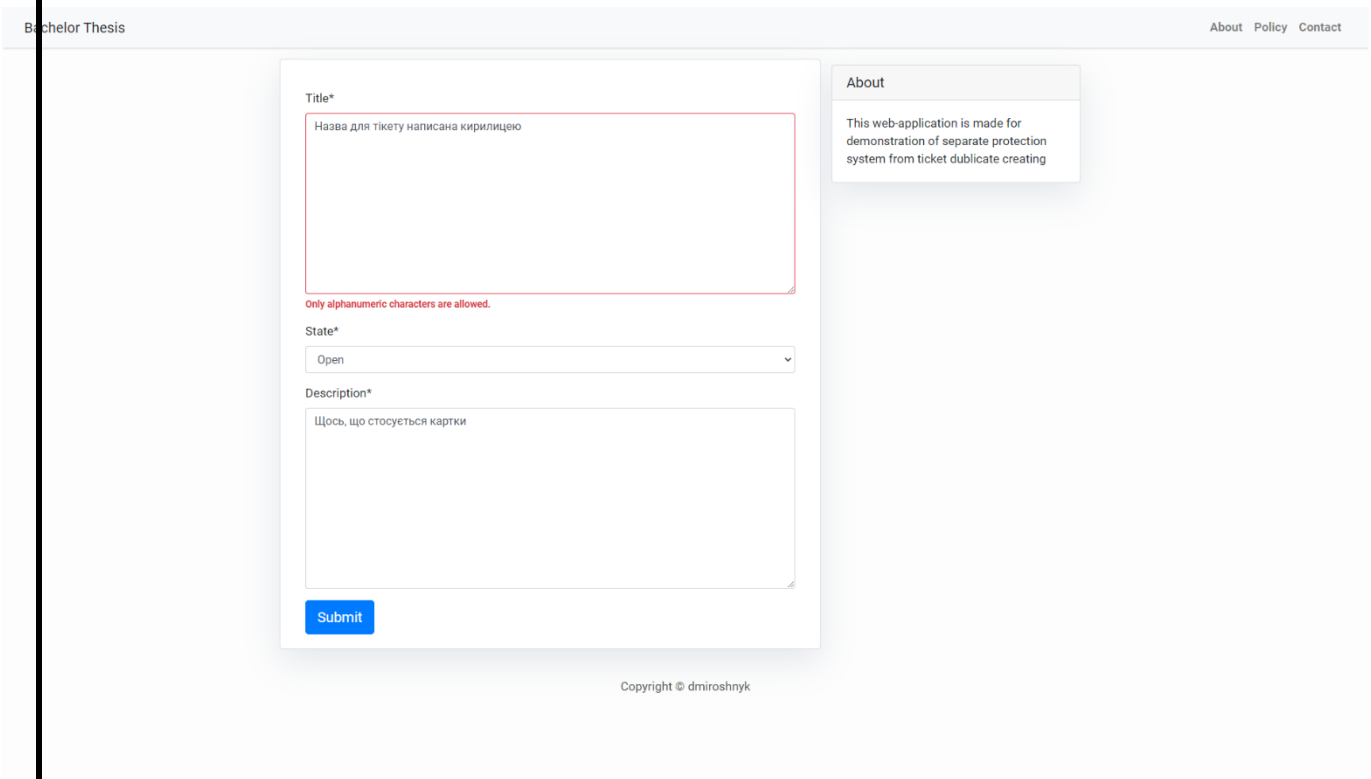


Рисунок 4.4 Форма створення нового тікету

Створивши тікет з описом, який схожий на опис одного, або декількох з уже існуючих тікетів Ви отримаєте велике значення в таблиці, виражене у відсотках. Якщо Ви створюєте перший тікет, то одразу ж повернетесь на домашню сторінку зі списком, в якому, відповідно, буде лише одна картка.

В таблиці, яка показує оцінку схожості описів, можна знайти посилання на знайдені тікети. По ним можна перейти та повернутися, при цьому не втративши попередньо заповнену форму нового тікета. Таблиця дасть автору змогу переглянути найбільш схожі тікети та вирішити: зберігати нову картку чи ні.

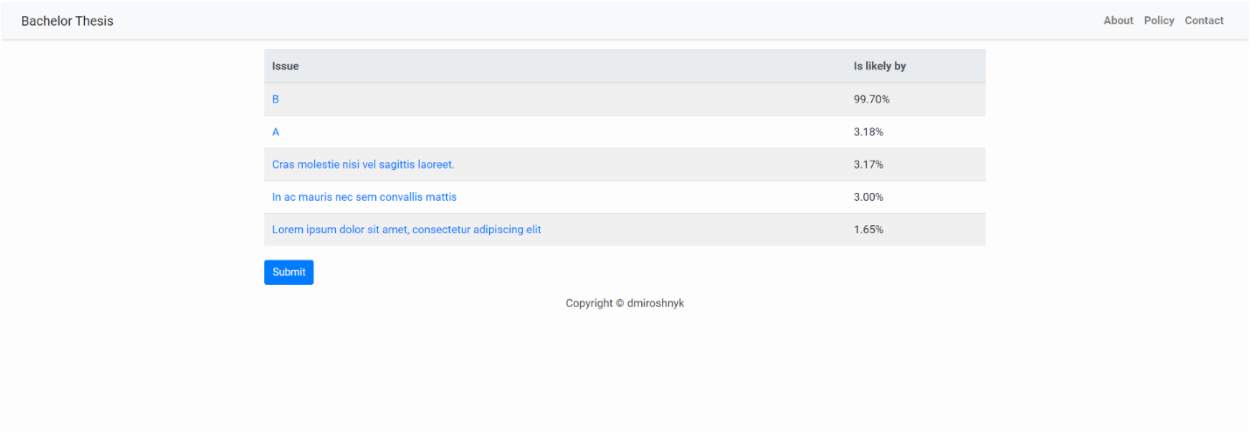


Рисунок 4.5 Таблиця з результатами оцінки нового тікета

Висновок до розділу 4

В четвертому розділі було описано розроблені компоненти веб-сервісу. IssueTracking компонента містить логіку обробки запитів та формування відповідей. DuplicationDetection – це компонент, який реалізує оцінку подібності текстів та надає інтерфейс доступу для основного модуля.

Інструкція користувача, описана у цьому розділі, демонструє, що інтерфейс зрозумілий та легкий у застосуванні, передбачає зберігання прогресу етапів створення тікетів.

ВИСНОВОК

Результатом бакалаврського дипломного проекту є веб-сервіс для відслідковування тікетів з використанням системи захисту від створення дублікатів.

Застосунок надає користувачу зрозумілий інтерфейс для створення і редагування тікетів. Навігація по веб-сервісу зручна та інтуїтивна для користувача, зберігає прогрес різних етапах роботи з тікетами.

Для написання системи були використані сучасні інструменти: веб-фреймворк Django на мові програмування Python. Базою даних у цьому сервісі є SQLite. Сервіс має три компоненти: основну, яка містить конфігурації, IssueTracking, яка описує логіку обробки запитів та DuplicationDetection компоненту, яка реалізує систему оцінки подібності текстів.

Компонент DuplicationDetection є самостійним, а тому може бути вилучений і застосований в інших сервісах, аналогічних приведеному у дипломному проекті.

Було розглянуто декілька способів оцінки подібності текстів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. В.Є. Бахрушин «Методи аналізу даних». Запоріжжя, Класичний приватний університет, 2011, с. 117
2. D.V. Lande, A.A. Snarskii, E.V. Yagunova, and E. Pronoza, “The Use of Horizontal Visibility Graphs to Identify the Words that Define the Informational Structure of a Text”, In: Proceedings of the 12th Mexican International Conference on Artificial Intelligence, 2013, с. 209-215.
3. С.І. Альперт Основні міри подібності та нові підходи до їх застосування при класифікуванні гіперспектральних космічних зображень Науковий Центр аерокосмічних досліджень Землі ІГН НАН України, м. Київ, Україна с. 145
4. Susan T. Dumais Latent semantic analysis [Електронний ресурс]. Режим доступу: https://asistdl.onlinelibrary.wiley.com/doi/full/10.1002/aris.1440380105 (дата звертання 15.02.2020)
5. David M. Blei, Andrew Y. Ng, Michael I. Jordan Latent Dirichlet allocation
6. Yoshua Bengio Learning Deep Architectures for AI. Dept. IRO, Universit´e de Montr´eal C.P. 6128, Montreal, Qc, H3C 3J7, Canada
7. Signature verification using a "Siamese" Time delay neural network, Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Sickinger and Roopak Shah AT&T Bell Laboratories Holmdel
8. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Jacob Devlin Ming-Wei Chang Kenton Lee
9. SQLite [Електронний ресурс]. Режим доступу: https://www.uk.w3ki.com/sql_certificate/ (дата звертання 12.05.2020)

ДОДАТОК 1

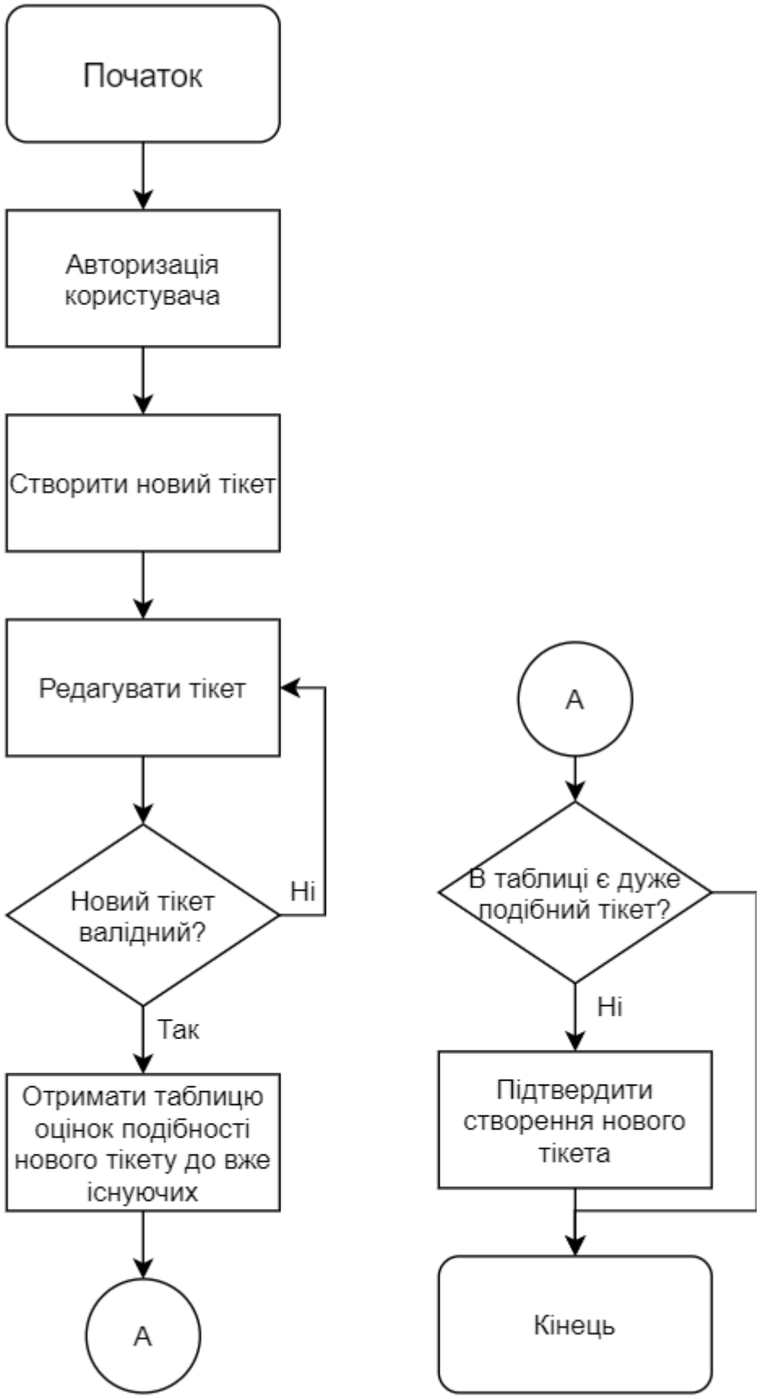
Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення

Схема принципова – схема алгоритму

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2020 р



					ІЛАЦ.467100.004 Д1		
Зм.	Арк.	№ документу	Підпис	Дата			
Розробив		Мірошник Д.В			Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Порєв В.М.					
Н. Контр.		Симоненко В.П.			Схема принципова – схема алгоритму	НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, гр. ІП-62	
Затверд.							

ДОДАТОК 2

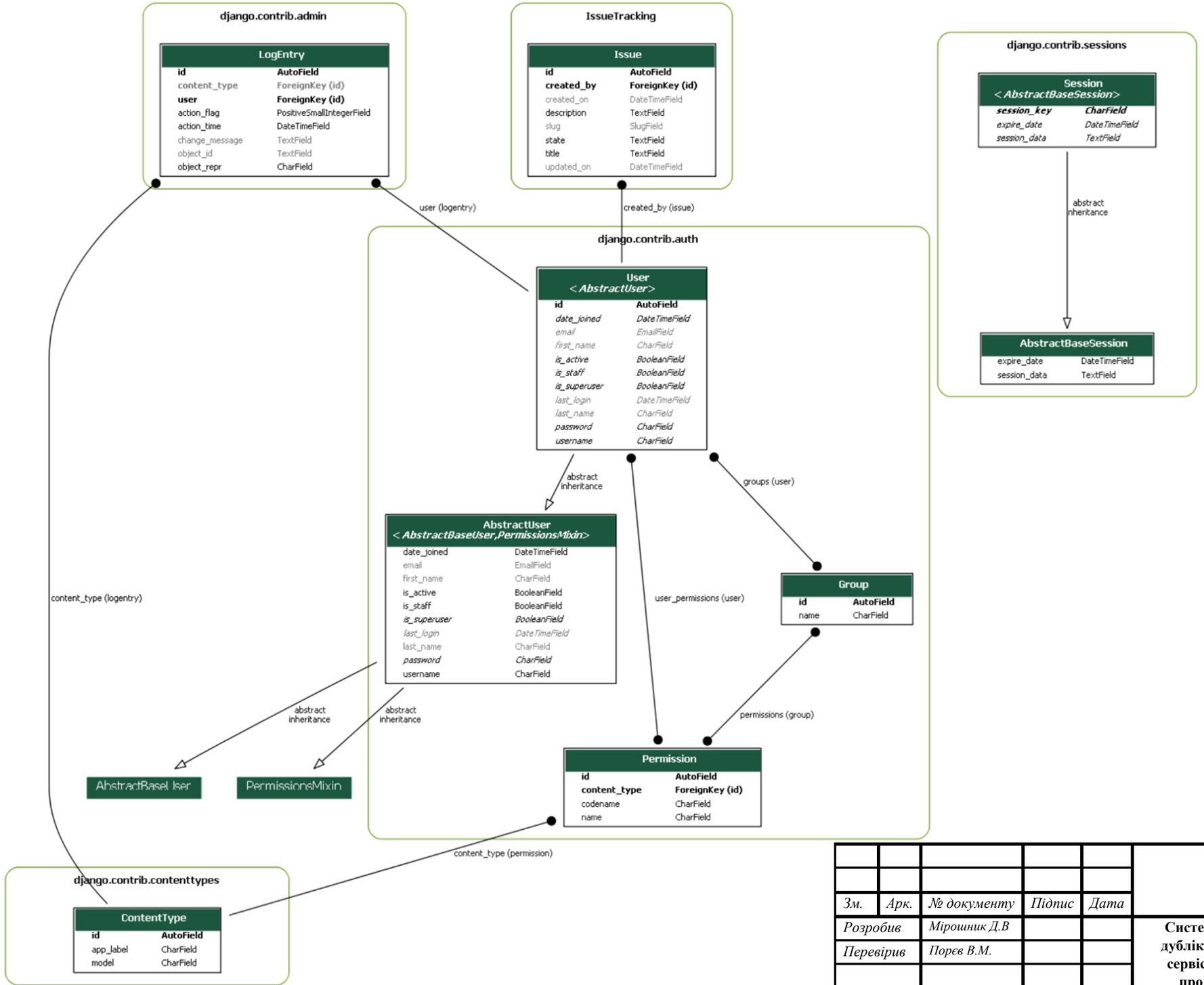
Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення

Схема функціональна – об'єктно-реляційне відображення

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2020 р



					ІЛАЦ.467100.005 Д2		
Зм.	Арк.	№ документу	Підпис	Дата			
Розробив		Мірошник Д.В			Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення (Схема функціональна – об'єктно-реляційне відображення)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Порєв В.М.					
Н. Контр.		Симоненко В.П.				НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, гр. ІП-62	
Затверд.							

ДОДАТОК З

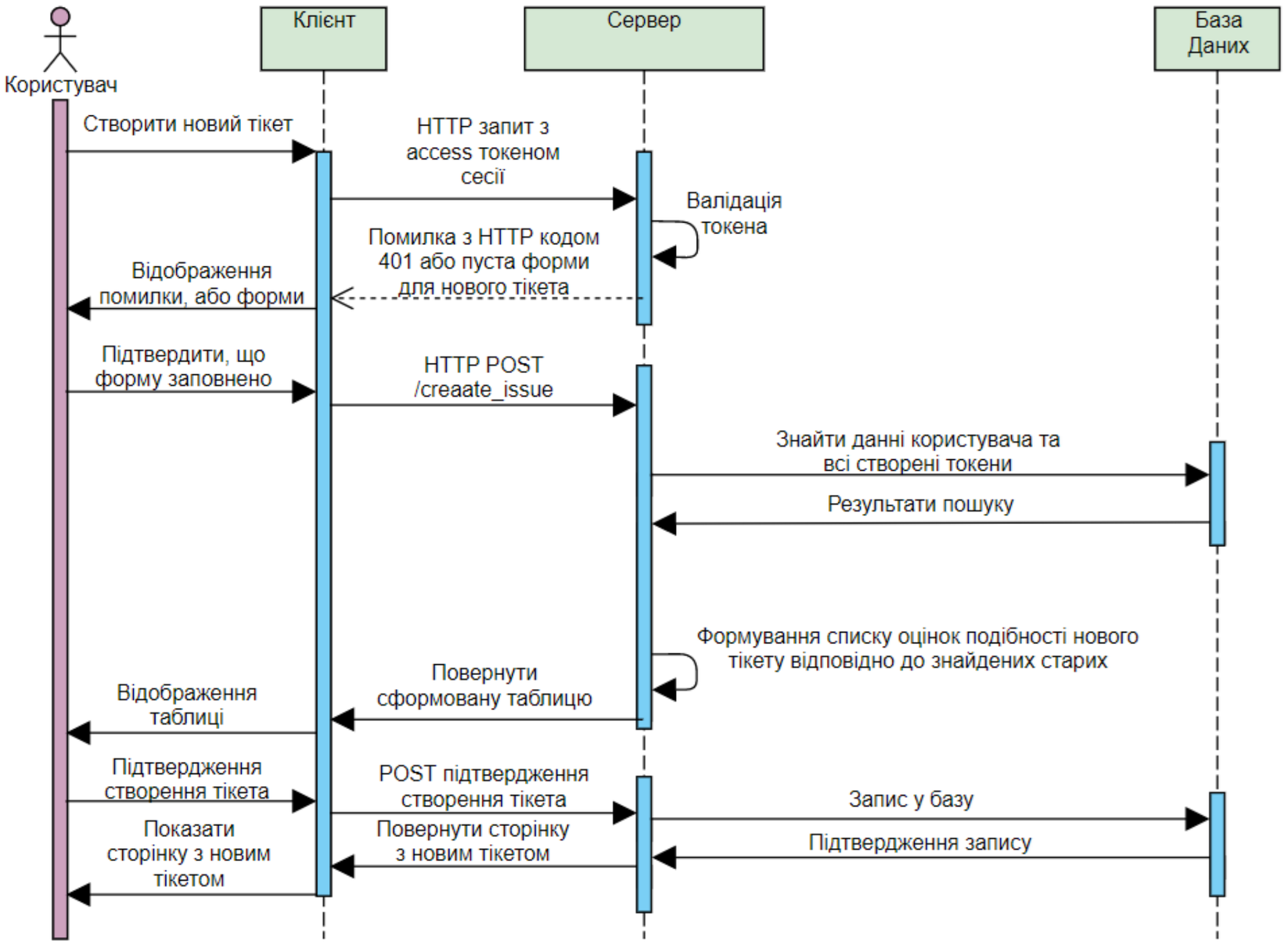
Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення

Схема структурна – діаграма послідовності

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ 2020 р



					ІЛАЦ.467100.006 ДЗ	
Зм.	*Арк.*	*№ документу*	*Підпис*	*Дата*		
Розробив		*Мірошник Д.В*			**Система захисту від створення дублікатів задач у виробничому сервісі менеджменту розробки програмного забезпечення (***Схема структурна – діаграма послідовності)*	Літ. / Аркуш: 1 / Аркушів: 1
Перевірив		*Порєв В.М.*				
Н. Контр.		*Симоненко В.П.*			*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, гр. ІП-62*	
Затверд.						